Chapter 1

MANAGING COMMUNICATIVE INTENTIONS WITH COLLABORATIVE PROBLEM SOLVING*

Nate Blaylock, James Allen and George Ferguson
Department of Computer Science
University of Rochester
Rochester, New York, USA
{blaylock,james,ferguson}@cs.rochester.edu

Abstract Dialogue systems need to be able to understand a user's communicative intentions, reason with those intentions, form their own communicative intentions, and realize those intentions with actual language to be uttered to the user. Oftentimes in dialogue systems, however, what these communicative intentions actually correspond to is never clearly defined. We propose a descriptive model of dialogue, based on collaborative problem solving, which defines communicative intentions as attempts to modify a shared collaborative problem-solving state between the user and system. Modeling dialogue at the level of collaborative problem solving allows us to model a wider array of dialogue types than previous models, including the range of collaboration paradigms (master-slave to mixed-initiative) and interaction types (planning, execution, and interleaved planning and execution). It also provides a definition for utterance-level communicative intentions for use within a dialogue system.

Keywords: Communicative Intentions, Collaborative Problem Solving, Dialogue Systems

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

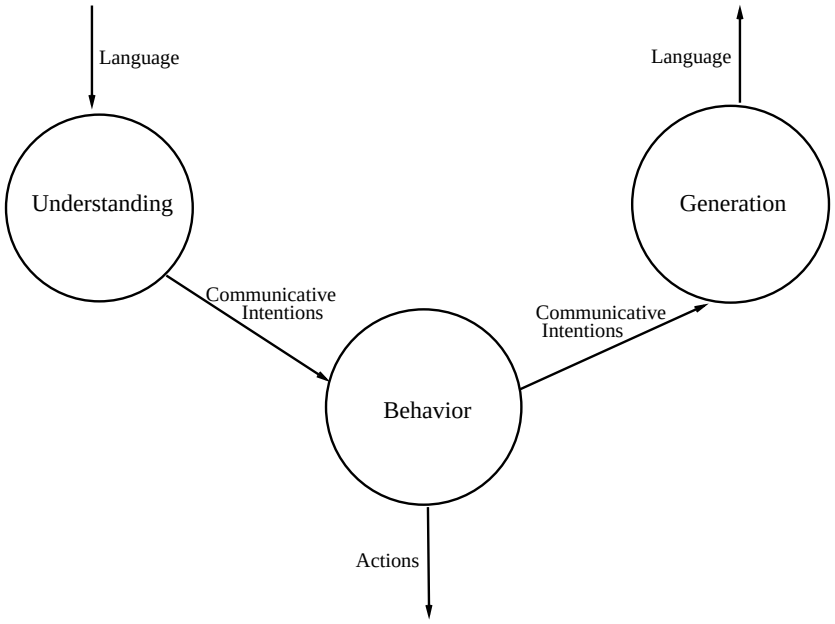

Figure 1.1. Conceptual Subsystems of a Conversational Agent

Introduction

Language does not occur in a vacuum. It is a means to an end. This work is done in the context of the goal of producing a *conversational agent*: a dialogue system which can interact in natural language (i.e., conversational) as well as plan and act to affect the world (i.e., an agent).

A conversational agent's functionality can be divided into three areas (as shown in figure 1.1): *interpretation*, *behavior*, and *generation* (Allen et al., 2001a). In interpretation, the system must be able to *understand* what is meant by a user's utterance. This goes much deeper than just 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? These conversational "ends" are known as *communicative intentions*.

In behavior, the system reasons with the user's communicative intentions[1] to decide what action to take in the world, as well as to form its own intentions for communication with the user.

[1] And possibly other things such as world state, its own goals and intentions, etc.

In generation, the system's communicative intentions are converted into natural language and uttered to the user.

The tasks of interpretation and generation are typically divided into many levels of analysis (speech recognition/synthesis, morphology, syntax, semantics, pragmatics, etc.), and an enormous body of research has been done at all of them. For the present undertaking, we concern ourselves only with the topmost level, which we call the *intention/language interface*, where communicative intentions are converted to and from a high-level semantic form (i.e., communicative acts). The process of inferring communicative intentions from communicative acts is *intention recognition* and the reverse process (converting communicative intentions into communicative acts) is *content planning*.

While communicative intentions are central to all these processes, what they actually correspond to is typically never defined within dialogue systems. We introduce a descriptive model of dialogue based on collaborative problem solving to define communicative intentions as attempts to modify a shared collaborative problem-solving state between the user and system. This model defines both what communicative intentions are and their effects at the problem-solving level.

Collaborative problem solving (CPS) is the process of agents jointly choosing goals, planning and acting in order to accomplish them. There are several advantages to modeling dialogue with collaborative problem solving. First, modeling *problem solving* instead of just planning or execution extends model coverage to dialogues involving both planning *and* execution. Similarly, modeling *collaboration* takes into account the joint nature of dialogue, allowing us to model the range of collaboration paradigms, from master-slave to mixed-initiative, as well as negotiation subdialogues. In addition, modeling communicative intentions at this level can free the behavior reasoning component in a dialogue system from having to worry about linguistic issues, allowing it to concentrate solely on reasoning about collaborative problem solving.

In the remaining sections, we first discuss previous work which has been done in this area. We then describe our descriptive model of dialogue in detail, showing how communicative intentions are defined within the model. We then exemplify the model with several dialogue examples. After that, we discuss how our model is used within the TRIPS dialogue system. Finally, we end with some conclusions and a discussion of future work.

1. Previous Work

Previous work falls roughly into two areas: models of collaborative planning, which look at how agents build joint plans, and models of dialogue. We discuss these below.

1.1 Models of Collaborative Planning

While much work has been done on formalizing joint plans and activity (Searle, 1990; Levesque et al., 1990; Grosz and Kraus, 1996), relatively little has looked at the collaboration that takes place between agents that are jointly planning and/or acting. One line of research that models collaboration is the SharedPlan formalism.

The SharedPlan formalism (Grosz and Kraus, 1996) was created in part to explain the intentional structure of discourse (Grosz and Sidner, 1986; 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 building SharedPlans.

- *Select_Rec*: An individual agent selects a recipe to be used to attain a given subgoal.
- *Elaborate_Individual*: An individual agent decomposes a recipe into (eventually) completely specified atomic actions.
- *Select_Rec_GR*: Intuitively, the same as Select_Rec, only at the multi-agent level.[2] A group of agents select a recipe for a subgoal.
- *Elaborate_Group*: The multi-agent equivalent of Elaborate_Individual — a group of agents decompose a recipe.

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

Lochbaum (Lochbaum, 1998) developed an intention recognition algorithm, based on the SharedPlan formalism, which models 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.

Another system based on SharedPlans is the COLLAGEN system (Rich et al., 2001), an intelligent user-interface system which models

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

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 "utterances" based on the current state of the dialogue. COLLAGEN gives options such as "Let's do X" to decide on a task, "How do I...?" for decomposing a recipe further, and "Where are we?" for summarizing where the user is in the task. Because user options are based on the collaborative context, the system is able to interact with the user in a more natural and intelligent way.

Shortcomings of SharedPlans. 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, building a conversational agent, there are several shortcomings in this model.

First, SharedPlans only models collaboration for joint 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 involving (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 collaboration at an utterance-by-utterance level (which is needed to represent communicative intentions). As an example, consider the *Elaborate_Group* operator, which 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. An *Elaborate_Group* can and often does consist of many individual utterances. In order to build a dialogue system, we need to be able to model the communicative intentions behind a single utterance.

We believe that our model may be compatible with SharedPlans and can be seen as specifying the details of the SharedPlan operators at an utterance level (see section 2.4 below).

1.2 Models of Dialogue

Much work has been done on modeling dialogue based on plans. Recent work (Ramshaw, 1991; Lambert and Carberry, 1991; Carberry et al., 1992; Chu-Carroll and Carberry, 2000) models dialogue with plans at several levels. While the types of levels differ with each model, all models include at least a domain level and a problem-solving level. In these models, user utterances are interpreted as corresponding to certain actions within recipes at the various levels.

Problem-solving level recipes are meta-plans which specify how domain plans are built. For example, problem-solving recipes in (Carberry et al., 1992) include the recipe *Make-Plan*, which contains three steps: 1) *Explore-Recipes-for*, 2) *Choose-Best-Specialization-Of*, and 3) *Build-Plan*. *Build-Plan*, in turn, executes *Make-Plan* for all the steps of the chosen recipe.

Multi-level dialogue models have greatly increased the types of dialogue that can be covered in dialogue systems. There are, however, a few areas in which we would like to expand that coverage.

With the exception of (Chu-Carroll and Carberry, 2000), previous work in intention recognition has only modeled master-slave collaboration (cf. (Grosz and Sidner, 1990)). Most previous research was restricted to information-seeking dialogues 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. 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.

Another area where we need more coverage is the same that was discussed in section 1.1 above: modeling dialogue involving both planning *and* acting. Previous work makes 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 1.1, in the domains we are interested in, collaboration involves

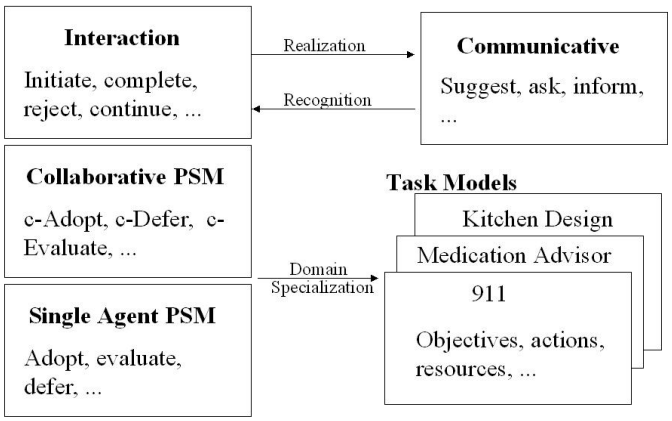

Figure 1.2. The Collaborative Problem-Solving Model

both acting and planning, often interleaving the two. We need a model of collaborative problem solving which covers all of these cases.

2. A Collaborative Problem-Solving Model

This section describes our preliminary work in building a collaborative problem-solving (CPS) model to be used as a descriptive model of dialogue (Blaylock, 2002; Allen et al., 2002). A graphical representation of the model is shown in figure 1.2. On the right are several *task models* which specialize the model to specific domains. These contain specific knowledge about the domain such as what objectives are in the domain; what objects are considered resources; and knowledge about situations. They also include knowledge about actions and recipes in the domain.

A task model is used as a "plug-in" to the rest of the collaborative problem-solving model, which is domain independent. Objects in the task model are specializations of abstract objects in the (single-agent and collaborative) problem-solving models. These models are comprised of a set of acts, the execution of which updates the (single-agent or collaborative) *problem-solving state* of the agents, allowing agents to do things such as evaluating and adopting objectives and recipes, executing plans, and so forth.

At the collaborative level, it is impossible for an agent to single-handedly change the CPS state. Doing so involves the cooperation of

both[3] agents involved (cf. (Traum, 1994)). This means, in other words, that CPS acts are *not* executable by a single agents. How is a single agent to affect the CPS state, then? It must do so through *interaction* with the other collaborating agent.

Interaction acts are single-agent actions used to negotiate changes in the CPS state. If the agents are amenable to the change and cooperate, a certain set of interaction acts will result in the generation of a CPS act, which changes the CPS state. Interaction acts are realized by *communicative acts* in order to be uttered by an agent.

In the rest of this section, we describe the individual parts of the model in more detail. We then revisit previous work in light of the model make some comparisons.

2.1 Task Models

A task model contains domain specific knowledge that is used in the model. This is how domain independence is achieved for the model. The other levels besides the task model do not change from domain to domain. The CPS model utilizes abstract problem-solving objects which are specialized by different task models. We believe that collaborative problem solving remains the same, regardless of the task and domain (Allen et al., 2000; Allen et al., 2001b). If agents are collaborating about travel, medication, or kitchen design, the collaboration seems to remain the same at an abstract level.

A task model contains domain-specific specializations of the following abstract *problem-solving objects*.

- *Objectives*: The goals in the domain. For example, in a rescue domain, objectives could include rescuing a person, evacuating a city, and so forth.

- *Recipes*: An agent's beliefs of how to attain an objective. Although we do not adhere to any specialized definition of recipe, one example is Carberry's domain plan library (Carberry, 1990) which has action decomposition information about objectives. An agent's recipe library can be expanded or modified through (collaborative or single-agent) planning.

- *Atomic Actions*: Directly-executable actions in the domain.

[3]We only discuss the case of two collaborating agents. While we hope that the taxonomy we present will be extensible to multi-agent problem solving and dialogue, we leave this as a topic of future research.

- *Objects/Resources*: Objects in the world as well as which objects are considered resources for the domain. Resources are objects that are used in recipes.
- *Situations*: The state of the world (or a possible world). In all but the simplest domains, an agent may only have partial knowledge about a given situation.

These problem-solving (PS) objects are then used as arguments for PS acts, at the single-agent level, and CPS acts at the collaborative level.

2.2 Single-Agent Problem Solving

In order to see how agents collaborate to solve problems, it is instructive to first look at an abstract model of how a single-agent solves problems. The problem-solving (PS) 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., (Cohen and Levesque, 1990; Rao and Georgeff, 1991)), which use agent intentions, beliefs and so forth to predict agent action and changes in the agent's state. Our PS model describes possible changes in the agent's problem-solving state at the granularity at which collaboration occurs.

An agent's problem-solving activities occur in three phases. These need not proceed in order and may be freely interleaved.

- *Determining Objectives*: In this phase, an agent manages its objectives, deciding to which is it committed, which will drive its current behavior, etc.
- *Determining and Instantiating Recipes for Objectives*: In order to attain an objective, an agent chooses a recipe to use to work towards the objective. An agent may either choose and instantiate a recipe from its recipe library (see section 2.1 above), or it may choose to *create* a new recipe via planning.
- *Acting*: In this phase, an agent follows a recipe and executes atomic actions. This phase also includes monitoring the execution to check for success.

There are several things to note about this abstract, single-agent problem-solving model. First, our model does not impose any strict ordering on the phases above. An agent may begin executing a partially instantiated recipe and do more instantiation later as necessary. An agent may also perform some plan in order to help it in deciding what recipe to use for another objective. As we discuss in more detail

below, this allows the model to cover collaboration involving interleaved planning and acting.

In our model, problem-solving (PS) acts are the operators which an agent uses to affect its problem-solving state. These can be classified into two groups: acts relating to commitment and acts related to reasoning.

Acts Relating to Commitment. These acts change the agent's commitment to the various PS objects. The acts are listed here.[4]

- *Adopt*: Commits the agent to the PS object, i.e., the agent intends to do/use/pursue the object.
- *Select*: Moves an adopted object into the small set which is currently influencing the agent's behavior (i.e., in acting).
- *Defer*: Tables a selected object. The agent is still committed to the object, but it is no longer influencing the agent's current behavior.
- *Abandon*: Removes an agent's commitment to an object which has not yet fulfilled its purpose.
- *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. (Note that not all PS acts are applicable to all PS objects.)

Objectives: An agent can **adopt** an objective, which commits the agent to pursuing it. An agent may have any number of adopted objectives, but there is only a small subset is active, controlling the agent's current activity. **Selecting** an objective makes it active, meaning that the agent is currently actively pursuing it. An agent can demote an active objective back to the adopted state by **deferring** it. An agent can also, at any time, **abandon** an objective, dropping its commitment to it. Finally, when an agent believes that an adopted objective has been attained, it may **release** it. Note that an objective is not automatically released when it is attained. An agent must *believe* that it has been attained and then consciously release it.

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

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

Figure 1.3. Application of Commitment PS Acts to PS Objects

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.

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 executed 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 and resources. 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.

Figure 1.3 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.[5]

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 abandon it, etc.) The agent can also **modify** the objective in a certain way (make it more specific, change a parameter, etc.)

Recipes: These are treated very similarly to objectives. 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 for 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

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

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

Figure 1.4. Application of Reasoning PS Acts to PS Objects

to a recipe. It is not clear what it would mean to **modify** a resource.

Situations: In all but the most trivial domains, an agent will not know the entire situation. **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.

Figure 1.4 gives a summary of the applicability of reasoning PS acts to PS objects. Similar to figure 1.3, an `x` shows the applicability of acts to objects.

2.3 Collaborative Problem Solving

Collaborative problem solving is an extension of single-agent problem solving. The granularity of acts that we discussed above is the granularity at which collaboration occurs between agents. These acts are more overt in collaborative problem solving since agents must communicate and coordinate their reasoning and commitments in maintaining a CPS state between them.

At the CPS level are CPS acts which apply to the PS objects, paralleling the single-agent PS model. In order to distinguish acts at the two levels, we append a *c-* before CPS acts, creating *c-adopt*, *c-select*, *c-modify*, and so forth. CPS acts have similar intuitive meaning to those at the PS level and we will not redefine them here.[6] One important difference is that a single agent cannot single-handedly perform a CPS act. This requires the cooperation and coordination of both agents.

[6]Several formal models of intention (Levesque et al., 1990; Grosz and Kraus, 1996) have explored intentional differences between single-agent plans and group plans. This is outside the scope of this chapter.

CPS acts are generated by the individual *interaction acts* of each agent. 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).

An agent beginning a new proposal performs an *initiate*. In the case of successful generation of the CPS act, the proposal is possibly passed back and forth between the agents, being revised with *continues*, until both agents finally agree on it, which is signified by an agent *not* adding any new information to the proposal but simply accepting it with a *complete*; this generates the proposed CPS act.

At any point in this exchange, either agent can perform a *reject*, which causes the proposed CPS act to fail, and the CPS state to remain unchanged. This ability of either agent to negotiate and/or reject proposals allows our model to handle not just the master-slave collaboration paradigm, but the whole range of collaboration paradigms (including mixed-initiative). And, as we discussed above, the model also allows for dialogue involving planning, acting (execution), or interleaved planning *and* acting.

It also provides a tangible definition of communicative intentions, by correlating them with interaction acts. Thus, for each utterance, there is at least one interaction act (together with CPS argument) that corresponds to the communicative intentions of the speaker. This definition now allows us to specify both the communicative intentions of utterances as well as the effects they have on the ongoing negotiation of changes to the CPS state.

2.4 Comparison with Previous Work

As we mention in section 1.1, we believe that our model will be compatible with the SharedPlan formalism (Grosz and Kraus, 1996). In fact, one way of looking at our model is an elaboration of the SharedPlan operators at an utterance level. Adoption, evaluation, etc. of *objectives* actually resides at a level higher than the SharedPlan model, since SharedPlans assumes that a high-level goal has already been chosen. The adoption, evaluation, etc. of recipes is a further elaboration of the Select_Rec_GR operator. Most other acts, such as adopting resources, evaluating actions, etc. provide the details of the Elaborate_Group level.

Our work is similar to the problem-solving-level plans of the work on dialogue models (Ramshaw, 1991; Lambert and Carberry, 1991; Carberry et al., 1992; Chu-Carroll and Carberry, 2000). While those models

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

Figure 1.5. A Planning Dialogue

provided some sample problem-solving recipes, we are trying to enumerate the complete taxonomy of collaborative problem-solving activities.

3. Examples

In this section, we give several examples in order to illustrate this dialogue model and to show some of the range of dialogues it can be used to describe. The representation of interaction acts in the dialogues is somewhat abbreviated due to space limitations. The representation given here is sufficient, however, to exemplify interaction acts intended by each utterance.

3.1 A Planning Example

Figure 1.5 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, the corresponding interaction acts are listed below it.[7]

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.)

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

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

Figure 1.6. An Acting Dialogue

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.

3.2 An Acting Example

Figure 1.6 gives an example of a dialogue involving only acting from a Medication Advisor domain (Ferguson et al., 2002), where 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 2.2 that selecting an action means to start actually executing 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.

3.3 An Interleaved Planning and Acting Example

Figure 1.7 shows a dialogue that involves interleaved planning and acting. At this point in the dialogue, the user and system have already

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

Figure 1.7. Interleaved Planning and Acting Dialogue

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

Figure 1.8. Rejection Dialogue

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 to take the victim.

3.4 A Rejection Example

Figure 1.8 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. The ability to reject is an important feature of non-master-slave collaboration paradigms. This example and the next demonstrate the ability of our model to handle the range of collaboration paradigms, including mixed-initiative.

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

Figure 1.9. Negotiation Dialogue

3.5 A Negotiation Example

Figure 1.9 shows an example of negotiation within our dialogue model. This mixed-initiative exchange demonstrates not only the ability to reject, but also the ability to use that to make a counterproposal.

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. This shows the value of mixed-initiative dialogue since agents often have disparate, yet complementary knowledge, which, when brought to bear can result in a better solution than either agent could have come up with on its own.

4. Use in Dialogue Systems

In this section we describe the practical use of the CPS model we presented, including how we are currently using it within the TRIPS dialogue system.

We have presented a descriptive model of communicative intentions: interaction acts. As we discussed briefly in section 2, interaction acts are realized by communicative acts in order to be uttered (as illustrated in figure 1.2). We can now more precisely define intention recognition and content planning. Intention recognition is the recognition of interaction acts from communicative acts, and content planning is the reverse: converting interaction acts into communicative acts.

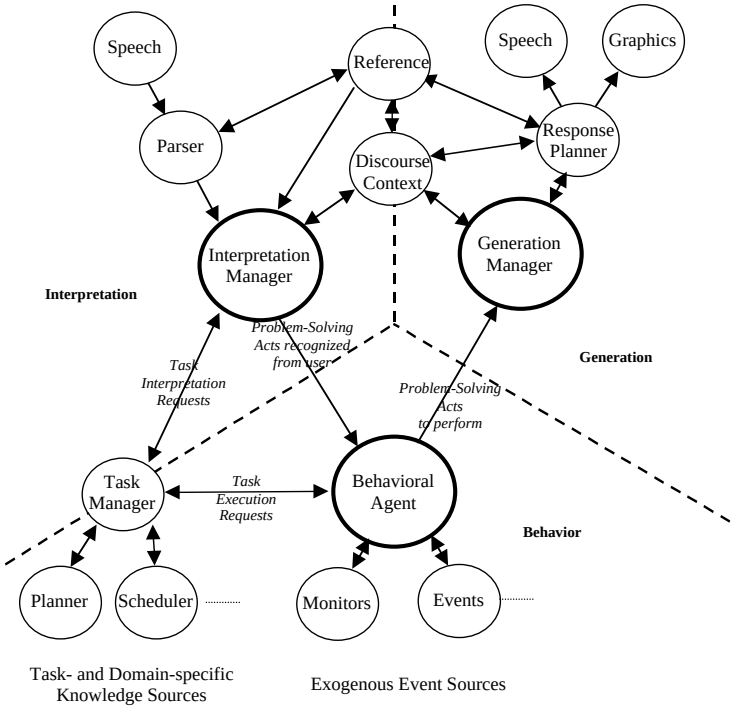

Figure 1.10. The TRIPS Architecture (Allen et al., 2001a)

Also, as we discussed above, modeling communicative intentions based on collaborative problem solving allows the behavioral reasoning component of a dialogue system to only worry about problem solving and not about linguistic issues. We hope this will allow for more complex system behavior as well as simpler behavioral component design (Blaylock et al., 2002b).

We are currently using the CPS model to represent communicative intentions within the TRIPS dialogue system (Ferguson et al., 2002; Blaylock et al., 2002a). The architecture of TRIPS is shown in figure 1.10. The details of the architecture are beyond the scope of this paper. The important thing to notice, however, is that there is no dialogue manager. Instead, the work of the dialogue manager has been divided into the three main components: the Interpretation Manager, the Behavioral Agent, and the Generation Manager. This is very similar to the conceptual subsystems of a conversational agent (figure 1.1) that we discussed in the introduction.

When an utterance is heard by the system, the Interpretation Manager coordinates interpretation (culminating with intention recognition) and passes the corresponding communicative intentions (represented as interaction acts) to the Behavioral Agent. The Behavioral Agent reasons with those communicative intentions (as well as its own goals and obligations and the state of the world) to determine the actions of the system. When the system decides to communicate with the user, the Behavioral Agent represents what it wants to communicate (its communicative intentions) as interaction acts and sends them to the Generation Manager which coordinates generation (beginning with content planning).

Within the system, intention recognition, behavioral reasoning on communicative intentions, and content planning based on the CPS model are all domain specific and still quite underdeveloped. We believe, however, that representing communicative intentions at the CPS level will enable us to build domain-independent components for all of these functions (see future work).

5. Conclusions and Future Work

We have presented a descriptive model of dialogue based on collaborative problem solving which defines communicative intentions and models a wider array dialogue than previous research. The model covers the range of collaboration paradigms (including mixed-initiative) and models dialogue involving (possibly interleaved) planning and acting.

As we mention above, this model currently lacks the level of formalism that other models have had (e.g., Levesque et al., 1990; Grosz and Kraus, 1996). For future work, we plan to formalize the model and explore similarities and differences with these other models.

We are currently using our model to annotate dialogues in various domains to ensure that the model is domain independent and to detect if there are any more collaborative problem-solving acts or problem-solving objects which need to be added.

We are currently developing a domain-independent intention recognition system based on the model (Blaylock, 2002). We are also interested in developing an agent that reasons with this model which would be used to drive system behavior, as well as in the content planning problem of converting interaction acts into communicative acts.

With a definition and representation of communicative intentions that correspond to utterances, we are interested in understanding and representing the difference between intention recognition (recognizing what the system was *intended* to recognize) and plan recognition (inferring more than that). These two concepts have often been conflated in previ-

ous research. As dialogue systems become more sophisticated, we believe it will be necessary to reason differently about what one was intended to understand, and anything addition which was gleaned from the evidence. Perhaps this could be useful in understanding and participating in non-collaborative or hostile dialogues.

References

Allen, J., Blaylock, N., and Ferguson, G. (2002). A problem solving model for collaborative agents. In *First International Joint Conference on Autonomous Agents and Multiagent Systems*, Bologna, Italy.

Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., and Stent, A. (2000). 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.

Allen, J., Ferguson, G., and Stent, A. (2001a). An architecture for more realistic conversational systems. In *Proceedings of Intelligent User Interfaces 2001 (IUI-01)*, pages 1–8, Santa Fe, NM.

Allen, J. F., Byron, D. K., Dzikovska, M., Ferguson, G., Galescu, L., and Stent, A. (2001b). Towards conversational human-computer interaction. *AI Magazine*, 22(4):27–37.

Ardissono, L., Boella, G., and Lesmo, L. (1996). Recognition of problem-solving plans in dialogue interpretation. In *Proceedings of the Fifth International Conference on User Modeling*, pages 195–197, Kailua-Kona, Hawaii.

Blaylock, N. (2002). Managing communicative intentions in dialogue using a collaborative problem solving model. Technical Report 774, University of Rochester, Department of Computer Science.

Blaylock, N., Allen, J., and Ferguson, G. (2002a). Synchronization in an asynchronous agent-based architecture for dialogue systems. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialog*, Philadelphia.

Blaylock, N., Dowding, J., and Allen, J. (2002b). A dialogue model for interaction with planners, schedulers and executives. In *Proceedings of the 3rd International NASA Workshop on Planning and Scheduling for Space*, Houston, Texas. To appear.

Carberry, S. (1990). *Plan Recognition in Natural Language Dialogue*. ACL-MIT Press Series on Natural Language Processing. MIT Press.

Carberry, S., Kazi, Z., and Lambert, L. (1992). Modeling discourse, problem-solving and domain goals incrementally in task-oriented dialogue. In *Proc. 3rd Int. Workshop on User Modeling*, pages 192–201. Wadern.

Chu-Caroll, J. and Brown, M. K. (1997). Initiative in collaborative interactions — its cues and effects. In Haller, S. and McRoy, S., editors, *Working Notes of AAAI Spring 1997 Symposium on Computational Models of Mixed Initiative Interaction*, pages 16–22, Stanford, CA.

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

Cohen, P. R. and Levesque, H. J. (1990). Intention is choice with commitment. *Artificial Intelligence*, 42:213–261.

Ferguson, G., Allen, J., Blaylock, N., Byron, D., Chambers, N., Dzikovska, M., Galescu, L., Shen, X., Swier, R., and Swift, M. (2002). The Medication Advisor project: Preliminary report. Technical Report 776, University of Rochester, Department of Computer Science.

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

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

Grosz, B. J. and Sidner, C. L. (1990). Plans for discourse. In Cohen, P. R., Morgan, J., and Pollack, M., editors, *Intentions in Communication*, pages 417–444. MIT Press, Cambridge, MA.

Lambert, L. and Carberry, S. (1991). A tripartite plan-based model of dialogue. In *Proceedings of the 29th ACL*, pages 47–54, Berkeley, CA.

Levesque, H., Cohen, P., and Nunes, J. (1990). On acting together. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 94–99, Boston. AAAI.

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

Lochbaum, K. E., Grosz, B. J., and Sidner, C. L. (2000). Discourse structure and intention recognition. In Dale, R., Moisl, H., and Sommers, H., editors, *Handbook of Natural Language Processing*, pages 123–146. Marcel Dekker, New York.

Ramshaw, L. A. (1991). A three-level model for plan exploration. In *Proceedings of the 29th ACL*, pages 39–46, Berkeley, CA.

Rao, A. S. and Georgeff, M. P. (1991). Modeling rational agents within a BDI-architecture. In Allen, J., Fikes, R., and Sandewall, E., editors, *Principles of Knowledge Representation and Reasoning*, pages 473–484, Cambridge, Massachusetts. Morgan Kaufmann.

Rich, C., Sidner, C. L., and Lesh, N. (2001). COLLAGEN: Applying collaborative discourse theory to human-computer interaction. *AI Magazine*, 22(4):15–25. Also available as MERL Tech Report TR-2000-38.

Searle, J. R. (1990). Collective intentions and actions. In Cohen, P. R., Morgan, J., and Pollack, M., editors, *Intentions in Communication*, pages 401–415. MIT Press, Cambridge, MA.

Traum, D. R. (1994). A computational theory of grounding in natural language conversation. Technical Report 545, University of Rochester, Department of Computer Science. PhD Thesis.