AN ALGORITHM FOR PLAN RECOGNITION IN COLLABORATIVE DISCOURSE*

Karen E. Lochbaum
Aiken Computation Lab
Harvard University
33 Oxford Street
Cambridge, MA 02138
kel@harvard.harvard.edu

ABSTRACT

A model of plan recognition in discourse must be based on intended recognition, distinguish each agent's beliefs and intentions from the other's, and avoid assumptions about the correctness or completeness of the agents' beliefs. In this paper, we present an algorithm for plan recognition that is based on the SharedPlan model of collaboration (Grosz and Sidner, 1990; Lochbaum *et al.*, 1990) and that satisfies these constraints.

INTRODUCTION

To make sense of each other's utterances, conversational participants must recognize the intentions behind those utterances. Thus, a model of *intended* plan recognition is an important component of a theory of discourse understanding. The model must distinguish each agent's beliefs and intentions from the other's and avoid assumptions about the correctness or completeness of the agents' beliefs.

Early work on plan recognition in discourse, e.g. Allen & Perrault (1980); Sidner & Israel (1981), was based on work in AI planning systems, in particular the STRIPS formalism (Fikes and Nilsson, 1971). However, as Pollack (1986) has argued, because these systems do not differentiate between the beliefs and intentions of the different conversational participants, they are insufficient for modelling discourse. Although Pollack proposes a model that does make this distinction, her model has other shortcomings. In particular, it assumes a master/slave relationship between agents (Grosz and Sidner, 1990) and that the inferring agent has complete and accurate knowledge of domain actions. In addition, like many earlier systems, it relies upon a set of heuristics to control the application of plan inference rules.

In contrast, Kautz (1987; 1990) presented a theoretical formalization of the plan recognition problem, and a corresponding algorithm, in which the only conclusions that are drawn are those that are "absolutely justified." Although Kautz's work is quite elegant, it too has several deficiencies as a model of plan recognition for discourse. In particular, it is a model of *keyhole recognition* — the inferring agent observes the actions of another agent without that second agent's knowledge — rather than a model of intended recognition. Furthermore, both the inferring and performing agents are assumed to have complete and correct knowledge of the domain.

In this paper, we present an algorithm for intended recognition that is based on the SharedPlan model of collaboration (Grosz and Sidner, 1990; Lochbaum *et al.*, 1990) and that, as a result, overcomes the limitations of these previous models. We begin by briefly presenting the action representation used by the algorithm and then discussing the type of plan recognition necessary for the construction of a SharedPlan. Next, we present the algorithm itself, and discuss an initial implementation. Finally, because Kautz's plan recognition algorithms are not necessarily tied to the assumptions made by his formal model, we directly compare our algorithm to his.

ACTION REPRESENTATION

We use the action representation formally defined by Balkanski (1990) for modelling collaborative actions. We use the term *act-type* to refer to a type of action; e.g. boiling water is an act-type that will be represented by boil(water). In addition to types of actions, we also need to refer to the agents who will perform those actions and the time interval over which they will do so. We use the term *activity* to refer to this type of information[1]; e.g. Carol's boiling water over some time interval (t1) is an activity that will be represented by ⟨boil(water),carol,t1⟩. Throughout the rest of this paper, we will follow the convention of denoting arbitrary activities using uppercase Greek letters, while using lowercase Greek letters to denote act-types. In

*This research has been supported by U S WEST Advanced Technologies and by a Bellcore Graduate Fellowship.

[1]This terminology supersedes that used in (Lochbaum *et al.*, 1990).

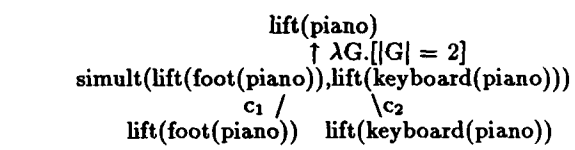
	Act-type	Activity
Relations	CGEN(γ_1,γ_2,C)	GEN(Γ_1,Γ_2)
	CENABLES(γ_1,γ_2,C)	ENABLES(Γ_1,Γ_2)
Constructors	sequence(γ_1,...,γ_n)	K(Γ_1,...,Γ_n)
	simult(γ_1,...,γ_n)	
	conjoined(γ_1,...,γ_n)	
	iteration($\lambda X.\gamma[X]$,{X_1,...X_n})	I($\lambda X.\Gamma[X]$,{X_1,...X_n})

Table 1: Act-type/Activity Relations and Constructors defined by Balkanski (1990)

addition, lowercase letters denote the act-type of the activity represented by the corresponding uppercase letter, e.g. γ = act-type(Γ).

Balkanski also defines act-type and activity *constructors* and *relations*; e.g. sequence(boil(water), add(noodles,water)) represents the sequence of doing an act of type boil(water) followed by an act of type add(noodles,water), while CGEN(mix(sauce,noodles), make(pasta_dish),C) represents that the first act-type *conditionally generates* the second (Goldman, 1970; Pollack, 1986). Table 1 lists the act-type and corresponding activity relations and constructors that will be used in this paper.

Act-type constructors and relations are used in specifying *recipes*. Following Pollack (1990), we use the term recipe to refer to what an agent knows when the agent knows a way of doing something. As an example, a particular agent's recipe for lifting a piano might be CGEN(simult(lift(foot(piano)), lift(keyboard(piano))), lift(piano), λG.[|G|=2]); this recipe encodes that simultaneously lifting the foot- and keyboard ends of a piano results in lifting the piano, provided that there are two agents doing the lifting. For ease of presentation, we will sometimes represent recipes graphically using different types of arrows to represent specific act-type relations and constructors. Figure 1 contains the graphical presentation of the piano lifting recipe.



↑C indicates generation subject to the condition C
c_i/ indicates constituent i of a complex act-type

Figure 1: A recipe for lifting a piano

The SharedPlan Augmentation Algorithm

A previous paper (Lochbaum *et al.*, 1990) describes an augmentation algorithm based on Grosz and Sidner's SharedPlan model of collaboration (Grosz and Sidner, 1990) that delineates the ways in which an agent's beliefs are affected by utterances made in the context of collaboration. A portion of that algorithm is repeated in Figure 2. In the discussion that follows, we will assume the context specified by the algorithm. SharedPlan*(G_1,G_2,**A**,T_1,T_2) represents that G_1 and G_2 have a *partial* SharedPlan at time T_1 to perform act-type **A** at time T_2 (Grosz and Sidner, 1990).

Assume:
Act is an action of type γ,
G_i designates the agent who communicates *Prop(Act)*,
G_j designates the agent being modelled
i, j $\in$ {1,2}, i $\neq$ j,
SharedPlan*(G_1,G_2,**A**,T_1,T_2).

...

4. Search own beliefs for Contributes(γ,**A**) and where possible, more specific information as to how γ contributes to **A**.

...

Figure 2: The SharedPlan Augmentation Algorithm

Step (4) of this algorithm is closely related to the standard plan recognition problem. In this step, agent G_j is trying to determine why agent G_i has mentioned an act of type γ, i.e. G_j is trying to identify the role G_i believes γ will play in their SharedPlan. In our previous work, we did not specify the details of *how* this reasoning was modelled. In this paper, we present an algorithm that does so. The algorithm uses a new construct: augmented rgraphs.

Augmented Rgraph Construction

Agents G_i and G_j each bring to their collaboration private beliefs about how to perform types of actions, i.e. recipes for those actions. As they collaborate, a significant portion of their communication is concerned with deciding upon the types of actions that need to be performed and how those actions are related. Thus, they establish mutual belief in a recipe for action[2]. In addition, however, the agents must also determine which

[2]Agents do not necessarily discuss actions in a fixed order (e.g. the order in which they appear in a recipe). Consequently, our algorithm is not constrained to reasoning about actions in a fixed order.

agents will perform each action and the time interval over which they will do so, in accordance with the agency and timing constraints specified by their evolving jointly-held recipe. To model an agent's reasoning in this collaborative situation, we introduce a dynamic representation called an *augmented recipe graph*. The construction of an augmented recipe graph corresponds to the reasoning that an agent performs to determine whether or not the performance of a particular activity makes sense in terms of the agent's recipes and the evolving SharedPlan.

Augmented recipe graphs are comprised of two parts, a *recipe graph* or *rgraph*, representing activities and relations among them, and a set of constraints, representing conditions on the agents and times of those activities. An rgraph corresponds to a particular *specification* of a recipe. Whereas a recipe represents information about the performance, in the abstract, of act-types, an rgraph represents more specialized information by including act-type performance agents and times. An rgraph is a tree-like representation comprised of (1) nodes, representing activities and (2) links between nodes, representing activity relations. The structure of an rgraph mirrors the structure of the recipe to which it corresponds: each activity and activity relation in an rgraph is derived from the corresponding act-type and act-type relation in its associated recipe, based on the correspondences in Table 1. Because the constructors and relations used in specifying recipes may impose agency and timing constraints on the successful performance of act-types, the rgraph representation is augmented by a set of constraints. Following Kautz, we will use the term *explaining* to refer to the process of creating an augmented rgraph.

AUGMENTED RGRAPH SCHEMAS

To describe the explanation process, we will assume that agents G_i and G_j are collaborating to achieve an act-type **A** and G_i communicates a proposition from which an activity Γ can be derived[3] (cf. the assumptions of Figure 2). G_j's reasoning in this context is modelled by building an augmented rgraph that explains how Γ might be related to **A**. This representation is constructed by searching each of G_j's recipes for **A** to find a sequence of relations and constructors linking γ to **A**. Augmented rgraphs are constructed during this search by creating appropriate nodes and links as each act-type and relation in a recipe is encountered.

By considering each *type* of relation and constructor that may appear in a recipe, we can specify general schemas expressing the form that the corresponding augmented rgraph must take. Table 2 contains the schemas for each of the act-type relations and constructors[4].

The algorithm for explaining an activity Γ according to a particular recipe for **A** thus consists of considering in turn each relation and constructor in the recipe linking γ and **A** and using the appropriate schema to incrementally build an augmented rgraph. Each schema specifies an rgraph portion to create and the constraints to associate with that rgraph. If agent G_j knows multiple recipes for **A**, then the algorithm attempts to create an augmented rgraph from each recipe. Those augmented rgraphs that are successfully created are maintained as possible explanations for Γ until more information becomes available; they represent G_j's current beliefs about G_i's possible beliefs.

If at any time the set of constraints associated with an augmented rgraph becomes unsatisfiable, a failure occurs: the constraints stipulated by the recipe are not met by the activities in the corresponding rgraph. This failure corresponds to a discrepancy between agent G_j's beliefs and those G_j has attributed to agent G_i. On the basis of such a discrepancy, agent G_j might query G_i, or might first consider the other recipes that she knows for **A** (i.e. in an attempt to produce a successful explanation using another recipe). The algorithm follows the latter course of action. When a recipe does not provide an explanation for Γ, it is eliminated from consideration and the algorithm continues looking for "valid" recipes.

To illustrate the algorithm, we will consider the reasoning done by agent Pam in the dialogue in Figure 3; we assume that Pam knows the recipe given in Figure 1. To begin, we consider the activity derived from utterance (3) of this discourse: $\Gamma_1 = \langle lift(foot(piano)), \{joe\}, t1 \rangle$, where t1 is the time interval over which the agents will lift the piano. To explain Γ_1, the algorithm creates the augmented rgraph shown in Figure 4. It begins by considering the other act-types in the recipe to which $\gamma_1 = lift(foot(piano))$ is related. Because γ_1 is a component of a simultaneous act-type, the simult schema is used to create nodes N1, N2, and the link between them. A constraint of this schema is that the constituents of the complex activity represented by node N2 have the same time. This constraint is modelled directly in the rgraph by creating the activity corresponding to *lift(keyboard(piano))* to have the same time as Γ_1. No information about the agent of this activity is known, however, so a variable, G_1, is used to represent the agent. Next, because the simultaneous act-type is related by a CGEN relation to *lift(piano)*, the CGEN schema is used to create node N3 and the link between N2 and N3. The first two constraints of the schema are satisfied by creating node N3 such that its activity's agent and time are the

[3] Γ need not include a complete agent or time specification.

[4] The technical report (Lochbaum, 1991) contains a more detailed discussion of the derivation of these schemas from the definitions given by Balkanski (1990).

Recipe	Augmented Rgraph	
	Rgraph	Constraints
CGEN(γ, δ,C)	$\langle \delta, G, T \rangle$ $\uparrow$ GEN Γ	G=agent(Γ) T=time(Γ) HOLDS'(C,G,T)
CENABLES(γ, δ,C)	$\langle \delta, G, T \rangle$ $\Uparrow$ ENABLES Γ	HOLDS'(C,agent(Γ),time(Γ)) BEFORE(time(Γ),T)
sequence($\gamma_1, \gamma_2, \ldots \gamma_n$)	$\mathbf{K}(\Gamma_1, \Gamma_2, \ldots, \Gamma_n)=\Delta$ $\mid c_i$ Γ_i	$\forall j$ BEFORE(time(Γ_j),time(Γ_{j+1})) agent(Δ)=$\bigcup_j$agent(Γ_j) time(Δ)=cover_interval({time(Γ_j)})[5]
conjoined($\gamma_1, \gamma_2, \ldots \gamma_n$)	$\mathbf{K}(\Gamma_1, \Gamma_2, \ldots \Gamma_n)=\Delta$ $\mid c_i$ Γ_i	agent(Δ)=$\bigcup_j$agent(Γ_j) time(Δ)=cover_interval({time(Γ_j)})
simult($\gamma_1, \gamma_2, \ldots \gamma_n$)	$\mathbf{K}(\Gamma_1, \Gamma_2, \ldots \Gamma_n)=\Delta$ $\mid c_i$ Γ_i	$\forall j$ time(Γ_j)=time(Γ_{j+1}) agent(Δ)=$\bigcup_j$agent(Γ_j) time(Δ)=cover_interval({time(Γ_j)})
iteration($\lambda X.\gamma[X]$, $\{X_1, X_2, \ldots X_n\}$)	$\mathbf{I}(\lambda X.\Gamma[X], \{X_1, \ldots X_n\})=\Delta$ $\mid c_i$ $[\lambda X.\Gamma[X]]X_i$	agent(Δ)=agent(Γ) time(Δ)=time(Γ)

Table 2: Rgraph Schemas

same as node N2's. The third constraint is instantiated and associated with the rgraph.

(1) Joe: I want to lift the piano.
(2) Pam: OK.
(3) Joe: On the count of three, I'll pick up this [deictic to foot] end,
(4) and you pick up that [deictic to keyboard] end.
(5) Pam: OK.
(6) Joe: One, two, three!

Figure 3: A sample discourse

Rgraph:

N3:{lift(piano),{joe} $\cup$ G_1,t1}
$\uparrow$ GEN
N2:**K**({lift(foot(piano)),{joe},t1},{lift(keyboard(piano)),G_1,t1})
$\mid c_1$
N1:{lift(foot(piano)),{joe},t1}

Constraints: {HOLDS'(λG.[|G| = 2],{joe} $\cup$ G_1,t1)}

Figure 4: Augmented rgraph explaining ⟨lift(foot(piano)),{joe},t1⟩

Merging Augmented Rgraphs

As discussed thus far, the construction algorithm produces an explanation for how an activity Γ is related to a goal **A**. However, to properly model collaboration, one must also take into account the context of previously discussed activities. Thus, we now address how the algorithm explains an activity Γ in this context.

Because G_i and G_j are collaborating, it is appropriate for G_j to assume that any activity mentioned by G_i is part of doing **A** (or at least that G_i believes that it is). If this is not the case, then G_i must explicitly indicate that to G_j (Grosz and Sidner, 1990). Given this assumption, G_j's task is to produce a coherent explanation, based upon her recipes, for how all of the activities that she and G_i discuss are related to **A**. We incorporate this model of G_j's task into the algorithm by requiring that each recipe have at most one corresponding augmented rgraph, and implement this restriction as follows: whenever an rgraph node corresponding to a particular act-type in a recipe is created, the construction algorithm checks to see whether there is already another node (in a previously constructed rgraph) corresponding to that act-type. If so, the algorithm tries to *merge* the augmented rgraph currently under construction with the previous one, in part by merging these two nodes. In so doing, it combines the information contained in the separate explanations.

The processing of utterance (4) in the sample dialogue illustrates this procedure. The activity derived from utterance (4) is Γ_2=⟨*lift(keyboard(piano))*, {*pam*}, *t1*⟩. The initial augmented rgraph portion created in explaining this activity is shown in Figure 5. Node N5 of the rgraph corresponds to the act-type *simult(lift(foot(piano)),lift(keyboard(piano)))* and includes information derived from Γ_2. But the rgraph (in Figure 4) previously constructed in explaining Γ_1 also includes a node, N2, corresponding to this act-type (and containing information derived from Γ_1). Rather than continuing with an independent explanation for Γ_2, the algorithm attempts to combine the information

[5]The function cover_interval takes a set of time intervals as an argument and returns a time interval spanning the set (Balkanski, 1990).

from the two activities by merging their augmented rgraphs.

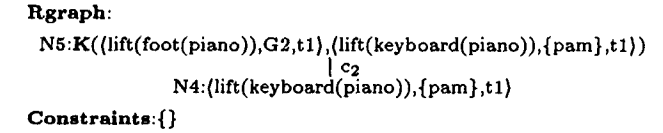


Figure 5: Augmented rgraph partially explaining ⟨lift(keyboard(piano)),{pam},t1⟩

Two augmented rgraphs are merged by first merging their rgraphs at the two nodes corresponding to the same act-type (e.g. nodes N5 and N2), and then merging their constraints. Two nodes are merged by unifying the activities they represent. If this unification is successful, then the two sets of constraints are merged by taking their union and adding to the resulting set the equality constraints expressing the bindings used in the unification. If this new set of constraints is satisfiable, then the bindings used in the unification are applied to the remainder of the two rgraphs. Otherwise, the algorithm fails: the activities represented in the two rgraphs are not compatible. In this case, because the recipe corresponding to the rgraphs does not provide an explanation for all of the activities discussed by the agents, it is removed from further consideration. The augmented rgraph resulting from merging the two augmented rgraphs in Figures 4 and Figure 5 is shown in Figure 6.

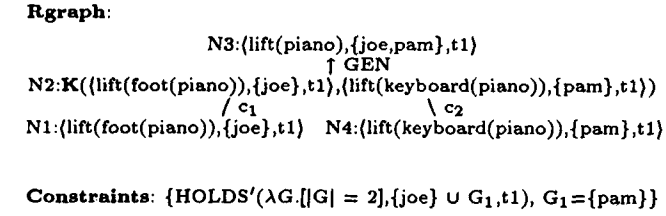


Figure 6: Augmented rgraph resulting from merging the augmented rgraphs in Figures 4 and 5

Implementation

An implementation of the algorithm is currently underway using the constraint logic programming language, CLP($\mathcal{R}$) (Jaffar and Lassez, 1987; Jaffar and Michaylov, 1987). Syntactically, this language is very similar to Prolog, except that constraints on real-valued variables may be intermixed with literals in rules and goals. Semantically, CLP($\mathcal{R}$) is a generalization of Prolog in which unifiability is replaced by solvability of constraints. For example, in Prolog, the predicate $X < 3$ fails if X is uninstantiated. In CLP($\mathcal{R}$), however, $X < 3$ is a constraint, which is solvable if there exists a substitution for X that makes it true.

Because many of the augmented rgraph constraints are relations over real-valued variables (e.g. the time of one activity must be before the time of another), CLP($\mathcal{R}$) is a very appealing language in which to implement the augmented rgraph construction process. The algorithm for implementing this process in a logic programming language, however, differs markedly from the intuitive algorithm described in this paper.

Rgraphs and Constraints vs. Egraphs

Kautz (1987) presented several graph-based algorithms derived from his formal model of plan recognition. In Kautz's algorithms, an explanation for an observation is represented in the form of an *explanation graph* or *egraph*. Although the term rgraph was chosen to parallel Kautz's terminology, the two representations and algorithms are quite different in scope.

Two capabilities that an algorithm for plan recognition in collaborative discourse must possess are the abilities to represent joint actions of multiple agents and to reason about hypothetical actions. In addition, such an algorithm may, and for efficiency should, exploit assumptions of the communicative situation. The augmented rgraph representation and algorithm meet these qualifications, whereas the egraph representation and algorithms do not.

The underlying action representation used in rgraphs is capable of representing complex relations among acts, including simultaneity and sequentiality. In addition, relations among the agents and times of acts may also be expressed. The action representation used in egraphs is, like that in STRIPS, simple step decomposition. Though it is possible to represent simultaneous or sequential actions, the egraph representation can only model such actions if they are performed by the same agent. This restriction is in keeping with Kautz's model of keyhole recognition, but is insufficient for modelling intended recognition in multiagent settings.

Rgraphs are only a part of our representation. Augmented rgraphs also include constraints on the activities represented in the rgraph. Kautz does not have such an extended representation. Although he uses constraints to guide egraph construction, because they are not part of his representation, his algorithm can only check their satisfaction locally. In contrast, by collecting together all of the constraints introduced by the different relations or constructors in a recipe, we can exploit interactions among them to determine unsatisfiability earlier than an algorithm which checks constraints locally. Kautz's algorithm checks each event's constraints independently and hence cannot determine satisfiability until a constraint is ground; it cannot, for example, reason that one constraint makes another unsatisfiable.

Because agents involved in collaboration dedicate a significant portion of their time to discussing the actions they need to perform, an algorithm for mod-

elling plan recognition in discourse must model reasoning about hypothetical and only partially specified activities. Because the augmented rgraph representation allows variables to stand for agents and times in both activities and constraints, it meets this criteria. Kautz's algorithm, however, models reasoning about actual event occurrences. Consequently, the egraph representation does not include a means of referring to indefinite specifications.

In modelling collaboration, unless explicitly indicated otherwise, it is appropriate to assume that all acts are related. In the augmented rgraph construction algorithm, we exploit this by restricting the reasoning done by the algorithm to recipes for **A**, and by combining explanations for acts as soon as possible. Kautz's algorithm, however, because it is based on a model of keyhole recognition, does not and cannot make use of this assumption. Upon each observation, an independent egraph must be created explaining all possible uses of the observed action. Various hypotheses are then drawn and maintained as to how the action might be related to other observed actions.

Conclusions & Future Directions

To achieve their joint goal, collaborating agents must have mutual beliefs about the types of actions they will perform to achieve that goal, the relations among those actions, the agents who will perform the actions, and the time interval over which they will do so. In this paper, we have presented a representation, augmented rgraphs, modelling this information and have provided an algorithm for constructing and reasoning with it. The steps of the construction algorithm parallel the reasoning that an agent performs in determining the relevance of an activity. The algorithm does not require that activities be discussed in a fixed order and allows for reasoning about hypothetical or only partially specified activities.

Future work includes: (1) adding other types of constraints (e.g. restrictions on the parameters of actions) to the representation; (2) using the augmented rgraph representation in identifying, on the basis of unsatisfiable constraints, particular discrepancies in the agents' beliefs; (3) identifying information conveyed in G_i's utterances as to *how* he believes two acts are related (Balkanski, 1991) and incorporating that information into our model of G_j's reasoning.

Acknowledgments

I would like to thank Cecile Balkanski, Barbara Grosz, Stuart Shieber, and Candy Sidner for many helpful discussions and comments on the research presented in this paper.

References

Allen, J. and Perrault, C. 1980. Analyzing intention in utterances. *Artificial Intelligence*, 15(3):143–178.

Balkanski, C. T. 1990. Modelling act-type relations in collaborative activity. Technical Report TR-23-90, Harvard University.

Balkanski, C. T. 1991. Logical form of complex sentences in task-oriented dialogues. In *Proceedings of the 29th Annual Meeting of the ACL, Student Session*, Berkeley, CA.

Fikes, R. E. and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208.

Goldman, A. I. 1970. *A Theory Of Human Action.* Princeton University Press.

Grosz, B. and Sidner, C. 1990. Plans for discourse. In Cohen, P., Morgan, J., and Pollack, M., editors, *Intentions in Communication.* MIT Press.

Jaffar, J. and Lassez, J.-L. 1987. Constraint logic programming. In *Proceedings of the 14th ACM Symposium on the Principles of Programming Languages*, pages 111–119, Munich.

Jaffar, J. and Michaylov, S. 1987. Methodology and implementation of a CLP system. In *Proceedings of the 4th International Conference on Logic Programming*, pages 196–218, Melbourne. MIT Press.

Kautz, H. A. 1987. *A Formal Theory of Plan Recognition.* PhD thesis, University of Rochester.

Kautz, H. A. 1990. A circumscriptive theory of plan recognition. In Cohen, P., Morgan, J., and Pollack, M., editors, *Intentions in Communication.* MIT Press.

Lochbaum, K. E., Grosz, B. J., and Sidner, C. L. 1990. Models of plans to support communication: An initial report. In *Proceedings of AAAI-90*, Boston, MA.

Lochbaum, K. E. 1991. Plan recognition in collaborative discourse. Technical report, Harvard University.

Pollack, M. E. June 1986. A model of plan inference that distinguishes between the beliefs of actors and observers. In *Proceedings of the 24th Annual Meeting of the ACL.*

Pollack, M. E. 1990. Plans as complex mental attitudes. In Cohen, P., Morgan, J., and Pollack, M., editors, *Intentions in Communication.* MIT Press.

Sidner, C. and Israel, D. J. 1981. Recognizing intended meaning and speakers' plans. In *Proceedings of IJCAI-81.*