MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

# Collaborating with Focused and Unfocused Users under Imperfect Communication

Neal Lesh, Charles Rich, Candace L. Sidner

TR-2000-39 November 2000

**Abstract**

A totally focused user always finishes the current task or subtask before moving on to another. Typical users, however, sometimes shift back and forth between incomplete tasks and do not always communicate before doing so. This behavior poses a problem for a software agent that uses plan recognition to support its collaboration with users. Our solution is a discourse interpretation algorithm which balances between asking too many questions about a user's intentions and sometimes being wrong about them.

*Submitted to 8th International Conference on User Modelling, Sonthofen, Germany, July 2001*

Submitted in November 2000.

# Collaborating with Focused and Unfocused Users under Imperfect Communication

Neal Lesh, Charles Rich and Candace L. Sidner

Mitsubishi Electric Research Laboratories

**Abstract.** A totally focused user always finishes the current task or subtask before moving on to another. Typical users, however, sometimes shift back and forth between incomplete tasks and do not always communicate before doing so. This behavior poses a problem for a software agent that uses plan recognition to support its collaboration with users. Our solution is a discourse interpretation algorithm which balances between asking too many questions about a user's intentions and sometimes being wrong about them.

## 1 Introduction

A key condition for successful task-oriented collaboration is that all the participants know which task or subtask is currently being worked on. In human collaboration, this condition is typically achieved through a combination of plan recognition, verbal communication, and para-linguistic cues, such as gesture, intonation, and facial expression.

For example, if two people are collaborating on a task whose steps are well known to both of them, it is quite natural for either collaborator to begin the next step without comment after completing the current step. On the other hand, if one of the collaborators decides to do something unexpected, she will often signal this intention by saying something like "Let's stop working on ...".

In earlier work [7], we described how a collaborative software agent could efficiently use plan recognition to infer users' intentions from their actions and utterances. However, the algorithm in [7] did not account for interruptions of the current task to work on a new task, or the role of communication in reducing ambiguity when unexpected focus shifts occur. These issues are particularly difficult for software agents, since they do not currently have access to the para-linguistic cues that facilitate natural human collaboration (and we have anecdotal evidence that people provide less task-level communication in general when interacting with a computer, even in spoken language).

Due to these effects, there is a tradeoff between a collaborative agent asking too many questions about a user's intentions and being wrong about them. The right balance depends on how often typical users unexpectedly shift their task focus and how often this intention is verbally communicated to the agent. In this paper, we present and analyze a discourse interpretation algorithm which is optimized for users who seldom make unexpected focus shifts and, when they do, verbally communicate their intention roughly half of the time.

### 1.1 Focused and Unfocused Users

The concept of *task focus* has been treated in the discourse and user interface literature mostly in terms of how to encourage problem solvers to be more focused. The basic idea of task focus is illustrated in Fig. 1.

Fig. 1 also illustrates the conventions we will use in figures in the rest of this paper. Lowercase letters, such as c, denote primitive actions. Uppercase letters, such as A, denote non-primitive actions. Underlined letters, such as c, denote completed actions. Each "snapshot" illustrates a discourse state comprised of a stack of goals, called the *focus stack* (the top goal on the focus stack is the current focus), and one or more plan trees (with unordered subplans). The indented execution trace at the far right is automatically produced by our discourse interpretation algorithm.

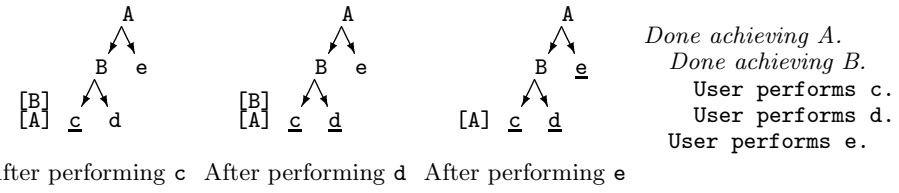



**Fig. 1.** A totally focused user always finishes the current task or subtask.

After performing c in Fig. 1, a totally focused user would next perform d. Then, since B is done, the next expected action is e, which entails an expected shift of focus back up to A. Typical users, however, are not totally focused. An example of an *unexpected focus shift* would be to perform e immediately after c. Starting and ending interruptions are also examples of unexpected focus shifts.

The possibility of unexpected focus shifts increases the likelihood of ambiguity, i.e., multiple possible explanations for an observed action. For example, Fig. 2 shows a variation of the plan tree in Fig. 1 in which step d has been added under A. Given this plan tree and the possibility of unexpected focus shifts, the second and third discourse states in Fig. 2 are both possible after performing d. In ambiguous situations like this, a collaborative agent needs to weigh the alternatives of explicitly asking the user which explanation is correct (which places a burden on the user) versus guessing the more likely, focused explanation (which means sometimes being wrong).

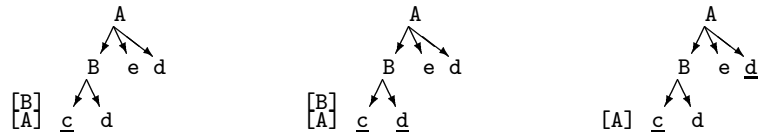



**Fig. 2.** Example of ambiguity between focused and unfocused explanations.

### 1.2 Imperfect Communication

Ambiguity such as in Fig. 2 can be avoided if the user communicates verbally about her intentions. In particular, an ideal user always communicates when she is about to make an unexpected focus shift. For example, if an ideal user's intentions were best represented by the unfocused explanation in Fig. 2, she would say something like the underlined utterance below:

*Achieving A.*
  *Paused in achieving B.*
    `User performs c.`
    `User says "Let's stop working on B."`
  `User performs d.`

If an agent knew that it was collaborating with an ideal user, it could, in the absence of communication to the contrary, safely ignore unfocused explanations. Unfortunately, due to the combination of missed and not-produced communication, our discourse interpretation algorithm must assume imperfect communication, i.e., that the typical user is not ideal.

## 2 Background and Terminology

This work reported takes place in the context of a larger effort to apply principles of human collaboration to human-computer interaction using the interface agent metaphor. Specifically, we have built an application-independent collaboration manager, called Collagen [10], based on the SharedPlan theory of task-oriented collaborative discourse [2, 9]. Collagen is implemented using Java Beans.

The issues raised in this paper are motivated by our experiences using Collagen to build intelligent assistants and tutors in a number of different application domains, including air travel planning [10], email [4], and industrial equipment operation. For example, Fig. 3 shows a tutoring interaction in which the user makes several unexpected focus shifts.

One of the most fundamental concepts in Collagen is a *discourse state*, which we formalize as a tuple $\langle \mathcal{S}, \mathcal{P} \rangle$, where the *focus stack*, $\mathcal{S}$, is a (possibly empty) stack of goals and $\mathcal{P}$ is a (possibly empty) set of (possibly incomplete) plans for these goals.[1] Given the current discourse state and an observation $\alpha$, which may be either a physical action or an utterance by either collaborator, *discourse interpretation* is the process of determining the new discourse state.

As we will see in more detail in the next section, discourse interpretation often involves both popping and pushing the focus stack and adding or extending plans in $\mathcal{P}$. Our earlier work [7] described a plan recognition algorithm, RECOGNIZE$(p, \alpha)$, which computes possible extensions to a plan $p$ which are consistent with the shared task model and include $\alpha$ as a next step. Also, RECOGNIZE$(\alpha)$ computes a set of new plans for top level goals which can start with $\alpha$. In this paper, we integrate plan recognition with maintenance of the focus stack to give a complete discourse interpretation algorithm.

[1] Grosz and Sidner [3] also postulate a third, linguistic, component of discourse state, which we do not address here.

```
   Done engaging the engine.
 1   Agent says "Let's practice engaging the engine."
 2   User says "Ok."
 3   Agent says "First, open the bleed valve."
 4   User opens the bleed valve.
     Done successfully opening the fuel valves.
 5     Agent says "Good. The next step is to open the fuel valves."
 6     User says "Ok."
       Paused in user opening fuel valve one.
 7       Agent says "First, you should open fuel valve one."
       Done user opening fuel valve two.
 8       User opens fuel valve two.
 9       Agent says "I wasn't expecting you to do that now, but it's ok."
       User continuing to open fuel valve one.
10       Agent says "Please return to opening fuel valve one."
11       User opens fuel valve one.
12       Agent says "Great. We have successfully opened the fuel values."
     Done starting the generator.
13     Agent says "Next, start the generator."
       Stopped interruption testing alarms.
14       User tests alarm one.
15       Agent says "Let's stop testing the alarms."
       Stopped interruption.
16       User purges fuel line.
17       Agent says "That wasn't relevant to engaging the engine."
18     User says "What next?"
19     Agent says "As I said, please start the generator."
20     User starts the generator.
21     Agent says "Good job. We have successfully engaged the engine."
```

**Fig. 3.** Segmented interaction history of a Collagen agent teaching the user how to operate a simulated gas turbine engine. Lines 8, 14 and 16 are examples of unexpected focus shifts by the user.

## 3 Discourse Interpretation Algorithm

Fig. 4 contains pseudo-code for the discourse interpretation algorithm used in Collagen. The essence of this algorithm is the preference order on types of explanations (possible next discourse states) for an observed action, expressed in the definition of EXPLAIN. These explanation types are defined in the following Case figures, which include both examples (illustrated in the style of Fig. 1) and pseudo-code which returns a (possibly empty) set $\mathcal{E}$ of discourse states.

Case 1 covers totally focused behavior, i.e., when there are no unexpected focus shifts. Case 1 is further broken down into three subcases: (a) there is no focus shift, i.e., the focus stack does not change, and (b) and (c), which are expected focus shifts to either the next subtask or a new task.

Our algorithm distinguishes Case 1a and Case 1b due to "lazy" popping of the focus stack. If we popped goals off the stack as soon as they were done, these two cases would have identical code. Instead, completed goals remain on the stack until interpretation of the next action. (In Fig. 1, note that B remains on the stack after performing d.) The reason for this approach is that a just-

completed goal may continue to be the topic of conversation, such as a tutor acknowledging a student's successful completion of the goal (see lines 12 and 21 in Fig. 3) or discussing whether the goal was done correctly.

The remaining four cases cover various types of unexpected focus shifts. Case 2 involves starting on a new subtask before completing the current subtask, thereby popping an incomplete goal (e.g., B) off the focus stack. Cases 3, 4 and 5 involve starting and ending interruptions. (There is also a variation on Case 3, which space does not allow us to include here, in which the stack is popped, i.e., the current task is abandoned, before starting the interruption.)

The example in Case 2 illustrates some of the generality with which plan recognition and updating the focus stack interact in our algorithm. Notice that

```
INTERPRET (⟨S, P⟩, α) ≡
  E ← EXPLAIN(⟨S, P⟩, α)
  if |E| = 1
    then return discourse state in E
    else
      ask questions to select a
        discourse state in E
      return selected discourse state
```

```
EXPLAIN (⟨S, P⟩, α) ≡
  return first non-empty set of:
    (1a) CURRENTTASK (⟨S, P⟩, α)
    (1b) NEXTTASK (⟨S, P⟩, α)
    (1c) NEWTASK (⟨S, P⟩, α)
    (2)  WITHINTASK (⟨S, P⟩, α)
    (4)  ENDINTERRUPTION (⟨S, P⟩, α)
    (3)  STARTINTERRUPTION (⟨S, P⟩, α)
    (5)  UNKNOWNGOAL (⟨S, P⟩, α)
```

**Fig. 4.** Discourse interpretation algorithm used in Collagen.

```
CURRENTSUBTASK (⟨S, P⟩, α) ≡
  E ← ∅
  if S is not empty
    p ← plan in P for top(S)
    foreach p' ∈ RECOGNIZE(p, α)
      P' ← P − p + p'
      E ← E + ⟨UPDATESTACK(S, p', α), P'⟩
  return E
```

```
UPDATESTACK (S, p, α) ≡
  push a sequence of subplans s_1...s_n
  onto S such that ∀i 1 < i ≤ n
    s_1 is a subplan of p for top(S),
    s_i is a subplan of p for s_{i−1},
    and α is a subplan of s_n.
  return S
```

**(a)** Continuing to work on the current subtask. (See performing d in Fig. 1 for example.)

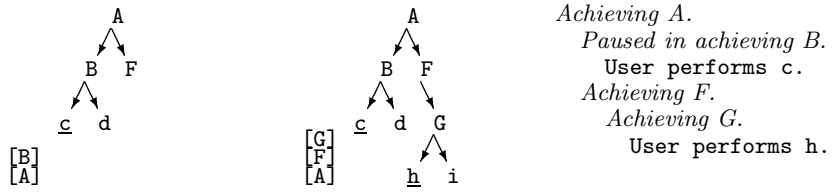
```
NEXTSUBTASK (⟨S, P⟩, α) ≡
  E ← ∅
  while top(S) is done
    pop(S)
    E ← E ∪ CURRENTSUBTASK(⟨S, P⟩, α)
  return E
```

**(b)** Working on the next subtask after completing the current subtask. (See performing e in Fig. 1 for example.)

```
NEWTASK (⟨S, P⟩, α) ≡
  E ← ∅
  if all goals on S are done
    S ← empty stack
    foreach p ∈ RECOGNIZE(α)
      E ← E + ⟨UPDATESTACK(S, p, α), {p}⟩
  return E
```

**(c)** Starting a new task after completing the current task.

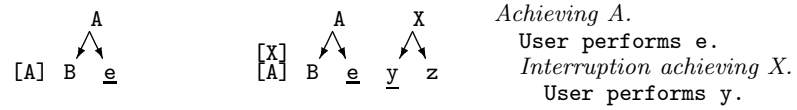
**Case 1.** No unexpected focus shifts.




Before performing h  After performing h

WITHINTASK $(\langle \mathcal{S}, P\rangle, \alpha) \equiv$
  **if** $\mathcal{S}$ is empty or top($\mathcal{S}$) is the root of it's plan
    **then** return $\emptyset$
  $\mathcal{E} \leftarrow$ CURRENTTASK$(\langle \mathcal{S}, P\rangle, \alpha) \cup$ NEXTTASK$(\langle \mathcal{S}, P\rangle, \alpha)$
  pop goals off $\mathcal{S}$ until top($\mathcal{S}$) is the root of it's plan
  **return** CURRENTTASK$(\langle \mathcal{S}, P\rangle, \alpha) - \mathcal{E}$

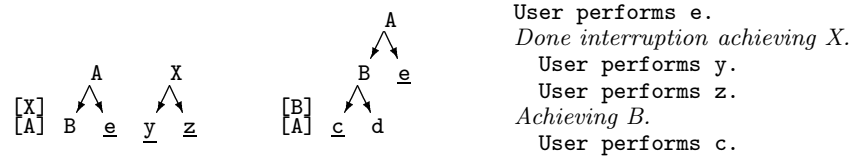
**Case 2.** Unexpected focus shift within the current task.




Before performing y  After performing y

STARTINTERRUPTION $(\langle \mathcal{S}, P\rangle, \alpha) \equiv$
  $\mathcal{E} \leftarrow \emptyset$
  pop all done goals off $\mathcal{S}$
  **foreach** $\langle \mathcal{S}', \mathcal{P}'\rangle \in$ NEWTASK$(\langle$empty stack$, \emptyset\rangle, \alpha)$
    $\mathcal{E} \leftarrow \mathcal{E} + \langle \mathcal{S} + \mathcal{S}', \mathcal{P} + \mathcal{P}'\rangle$
  **return** $\mathcal{E}$

**Case 3.** Interrupting incomplete current task by starting a known new task.




Before performing c  After performing c

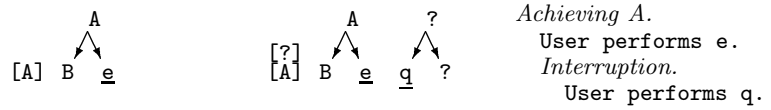
ENDINTERRUPTION $(\langle \mathcal{S}, P\rangle, \alpha) \equiv$
  **if** the root of the plan $p$ in $\mathcal{P}$ for top($\mathcal{S}$) is done
    pop all goals for $p$ off $\mathcal{S}$
    **return** first non-empty set of:
      CURRENTSUBTASK $(\langle \mathcal{S}, P\rangle, \alpha)$
      NEXTSUBTASK $(\langle \mathcal{S}, P\rangle, \alpha)$
      NEWTASK $(\langle \mathcal{S}, P\rangle, \alpha)$

**Case 4.** Returning to interrupted task after completing interruption.

after performing h, two goals (F and G) have been pushed on the stack, due to the fact that plan recognition has "interpolated" goal G.

Formally, an *interruption* is a goal that does does not contribute to (i.e., is not part of the plan tree for) the goal beneath it on the focus stack. Interruptions are the essential reason for adding a stack to the discourse interpretation algorithm in [7]. When an interruption is started, as in Case 3, the stack keeps track of the goal to return to when the interruption ends (Case 4).

Finally, Case 5, which always returns a non-empty set, takes care of when plan recognition fails to find an explanation for the observed action, e.g., because the task model is incomplete. Instead of simply generating no explanations, we found it more useful as a last resort to introduce an explicit unknown goal, to which any subsequent action may contribute.




Before performing q After performing q

UNKNOWNGOAL $(\langle \mathcal{S}, P \rangle, \alpha) \equiv$
pop all done goals off $\mathcal{S}$
create a plan $p$ for an unknown goal $g$ with subplan $\alpha$
**return** $\{\langle \text{push}(\mathcal{S}, g), \mathcal{P} + p \rangle\}$

**Case 5.** Interrupting incomplete current task by starting a unknown new task.

## 4 Analysis

In this section, we analyze our discourse interpretation algorithm with respect to the tradeoff between asking too many questions about a user's intentions and being wrong about them, and with respect to assumptions about the distribution of user behaviors.

### 4.1 Algorithmic Tradeoffs

The root of the tradeoff in discourse interpretation is ambiguity, i.e., when there is more than one possible explanation for an observation (see Fig. 2 for an example). When this happens, the algorithm's problem is to select the "correct" explanation, i.e., the explanation that reflects the user's actual intentions, from among the other "distracting" explanations. Our algorithm solves this problem by a mixture of preferences (guesses) based on explanation types (cases) and asking questions. Specifically (see Fig. 4), if there is a unique explanation in the first applicable case, we guess that explanation; we only ask questions to distinguish explanations when there is more than one explanation in the first applicable case.

| **Situations** | **(a)** | **(b)** | **(c)** | **(d)** | **(e)** |
|---|---|---|---|---|---|
| correct explanation | shift within (Case 2) | interruption (Case 3) | focused (Case 1) | shift within (Case 2) | interruption (Case 3) |
| distracting explanations | focused | focused or shift within | shift within or interruption | interruption | none |
| communication | no | no | n/a | don't care | don't care |
| **Behavior of Algorithm in Situation** | | | | | |
| old | incorrect | incorrect | correct | correct | incorrect |
| new | incorrect | incorrect | correct | correct | correct |
| conservative | questions | questions | questions | questions | correct |
| **Likelihood of Situation for User** | | | | | |
| ideal | zero | zero | high | low | low |
| typical | low | low | high | low | low |
| pathological | low | medium | medium | medium | medium |

**Table 1.** Analysis of algorithm and user behaviors in key situations.

Given the number of explanation types and the variability in user behavior, there are far too many possible forms of ambiguity to analyze in detail here. We have therefore selected five situations, defined in the top section of Table 1 to illustrate key aspects of the algorithmic tradeoffs. Each column in Table 1 corresponds to particular combination of correct explanation type, distracting explanation types and whether or not the user communicated about her unexpected focus shift, if any.

In the middle section of the table, we compare the "new" algorithm in this paper with the "old" algorithm in [7] and a "conservative" algorithm, which never guesses, but always asks questions when there is ambiguity. The only difference between the old algorithm and the new algorithm (in these five situations) is that the old algorithm does not handle interruptions (column (e)), which is unacceptable, because interruptions are part of natural collaborative behavior.

Notice that the new algorithm adopts an incorrect explanation in situations (a) and (b). In both of these situations, the user has made an unexpected focus shift which is ambiguous with other explanations that occur earlier in the algorithms preference order and has not communicated about it.

In contrast, the conservative algorithm is never wrong. However, the price of this perfection can be quite high. First of all, the number of questions that need to be asked is logarithmic in the number of distracting explanations. Furthermore, in a situation like (c), where the user is in fact being focused, questions about obscure alternative interpretations for her actions could be very disruptive to the flow of the collaboration.

### 4.2 Ideal, Typical and Pathological Users

Clearly, which algorithm or preference order on explanation types is best depends on the relative likelihood of the different situations, which in turn depends on

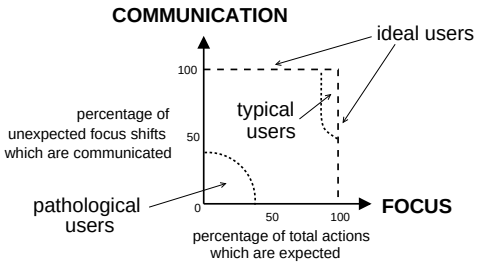



**Fig. 5.** Assumed distribution of users.

the characteristics of the users. The third section of Table 1 summarizes the likelihood of each situation for types of users in the distribution in Fig. 5.

*Ideal users*, although rare, never make unexpected focus shifts or, when they do, always communicate about them. Notice that the new algorithm does not penalize this type of user with incorrect explanations or irrelevant questions. *Typical users* stay focused about 90% of the time. We assume that about 5% of their actions are unexpected focus shifts within the current task (Case 2) and another 5% are interruptions (Case 3). Furthermore, we assume typical users communicate about 50% of their unexpected focus shifts. Of the remaining atypical users, we call *pathological users* those who unexpectedly shift focus up to half of the time and communicate about these shifts less than half of the time.

Given these assumptions, it is relatively straightforward to calculate the entries in the last section of Table 1. For example, the formula for the probability of situation (c) is:

$P_{\text{focused}} \times \overline{P}_{\text{distracting focused}}$
$\times (1 - \overline{P}_{\text{distracting shift within}} \times \overline{P}_{\text{distracting interruption}})$

Substituting values for the base probabilities above for typical users (.9, .1, .5, and .5, respectively) yields an answer of .6, which means that this situation is quite common. This result reinforces the argument against the conservative algorithm, because it means that this algorithm will often ask typical users a set of questions that were really only necessary half the time. Furthermore, notice that the likelihood of the only two situations in which the new algorithm is incorrect, namely (a) and (b), is low.

In summary the new algorithm is preferable to the conservative algorithm for both ideal and typical users, because it does not penalize them with irrelevant questions and rarely makes false guesses. Of course, as a user gets more pathological, the chances increase that the new algorithm's incorrect guesses will hamper its effectiveness.

## 5 Related Work and Conclusions

This work builds directly on our previous work [7] which described how to use plan recognition to efficiently extend mutually believed plans in a collaborative setting. In addition, our current work is close in spirit to other research on

plan recognition for cooperative dialogues. In particular, our use of the current discourse state to reduce potential ambiguities resembles Carberry's [1] focusing heuristics. Two contributions of this work are to give a detailed account of how to integrate this information into a discourse interpretation algorithm that allows for interruptions of the current task, and to give an analysis of the costs and benefits of preferring more focused explanations over less focused ones.

Previous work has also considered how to recognize the plans of someone who is performing more than one task at a time [5, 6]. These approaches assume simultaneous performance of multiple goals. In contrast, we believe that, in a collaborative setting, a more appropriate model is to be focused on one task at a time, and work on multiple goals by shifting focus back and forth.

Lochbaum [8] presented a plan recognition algorithm based on the Shared-Plan model of collaboration. Her plan recognizer does not chain recipes together (which ours does) and thus performs only "one level deep" recognition. Lochbaum does, however, make use of a wider range of relations by which actions contribute to goals than we do.

In conclusion, we have not only presented a specific discourse interpretation algorithm, but have also demonstrated an analysis methodology, which can be generally useful for collaborative software agents. In the future, it would be interesting to attempt to empirically validate the user distribution assumed in Fig. 5, and to work on how to recover from situations in which when our algorithm makes incorrect guesses.

## References


[1] S. Carberry. Incorporating default inferences into plan recognition. In *Proc. 8th Nat. Conf. AI*, volume 1, pages 471–8, july 1990.

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

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

[4] D. Gruen, C. Sidner, C. Boettner, and C. Rich. A collaborative assistant for email. In *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, Pittsburgh, PA, May 1999.

[5] H.A. Kautz and J.F. Allen. Generalized plan recognition. In *Proc. 5th Nat. Conf. AI*, pages 32–37, 1986.

[6] N. Lesh and O. Etzioni. A sound and fast goal recognizer. In *Proc. 14th Int. Joint Conf. AI*, pages 1704–1710, 1995.

[7] N. Lesh, C. Rich, and Candance L. Sidner. Using plan recognition in human-computer collaboration. In *Proc. of 7th Int. Conf. User Modeling*, pages 23–32, 1999.

[8] K. E. Lochbaum. An algorithm for plan recognition in collaborative discourse. In *Proc. 29th Annual Meeting of the ACL*, pages 33–38, Berkeley, CA, 1991.

[9] K. E. Lochbaum. A collaborative planning model of intentional structure. *Computational Linguistics*, 24(4), December 1998.

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