A Framework for Soliciting Clarification from Users During Plan Recognition

Robin Cohen
Computer Science
University of Waterloo
Waterloo, Ontario, Canada
rcohen@dragon.uwaterloo.ca

Ken Schmidt
Computing Science
University of Alberta
Edmonton, Alberta, Canada
kens@cs.ualberta.ca

Peter van Beek
Computing Science
University of Alberta
Edmonton, Alberta, Canada
vanbeek@cs.ualberta.ca

Abstract

In previous work, we used plan recognition to improve responses to users in an advice-giving setting. We then characterized when it was worthwhile to engage a user in clarification dialogue--the cases where plan ambiguity mattered to the formulation of a response. Our current research develops detailed algorithms for selecting what to say during the clarification dialogue. We propose a default strategy for selecting a clarifying question, together with a variety of options to reduce the length of the clarification dialogue. Each clarifying question is introduced in order to prune the set of possible plans. But the system will never overcommit in recognizing the user's plan and thus will never have to backtrack into a debugging dialogue with the user. In all, we now have a more precise formulation for what to say during clarification dialogues, and valuable characterizations of decisions which an advice-giving system must make in generating a dialogue with a user.

Introduction

In previous work, we examined the role of plan recognition in the generation of responses from advice-giving systems. We first presented an algorithm for critiquing possible plans of the user (van Beek 1987, van Beek and Cohen 1986), in order to provide clearer explanations for system responses. Simple "yes" or "no" responses were expanded to include reasons and suggestions for better alternatives, based on the user's profile.

Our next research effort (van Beek et al. 1993) studied more closely the effort expended in plan recognition, which typically attempts to determine the most likely plan of the user through heuristics (Allen 1979, Carberry 1983). Since these plan recognition systems sometimes had to backtrack from incorrect assumptions, we proposed a more cautious approach. All possible plans of the user were identified and then critiqued. Then, we asked whether it was necessary to identify the actual plan of the user, in order to provide an appropriate response. We discovered that in some cases, the same response would be generated, regardless of the specific user plan. Plan recognition systems could therefore be designed to stop processing when the ambiguity no longer mattered for the response being generated. Further, when it was clear that ambiguity about the user's plan had to be resolved, we proposed entering into a clarification dialogue with the user, in order to facilitate the process of plan recognition and to provide the most effective response for the user. This approach of involving the user in the processing was consistent with other papers of ours, which generally advocated more participation from users in certain automated reasoning systems (Cohen et al. 1990).

This earlier work made important progress on the general topic of clarification dialogues, suggesting one approach for when to initiate clarification in advice-giving settings and what to say in these dialogues. In order to gain insight into the criteria for clarification dialogues, we built a small implementation in the course advising domain, reported in (van Beek et al. 1993).

This paper goes beyond the previous work, focusing on the sub-topic of clarification in advice-giving systems. We have studied various alternatives for traversing the plan library, in order to generate appropriate questions during clarification dialogues. Our aim has been to develop algorithms which will reduce the length of these dialogues, either by asking fewer questions (identifying instances where certain questions which would have been proposed by our earlier algorithm were unnecessary) or by selecting key distinguishing questions more readily (in order to prune away part of the search space more effectively).

This paper therefore presents new insights into designing systems to include clarification dialogues, advancing improved algorithms for the generation of clarifying questions. These algorithms are employed in a context of plan recognition, continuing our earlier approach of critiquing plans and performing plan recognition to the point where ambiguity no longer matters. Through our work, the importance of including plans in user models and of directing the generation of explanations to specific user profiles is emphasized. In addition we now have a more precise formulation for what to say during clarification dialogues, and valuable characterizations of decisions which an advice-giving system must make in generating a dialogue with a user.

Background

In this section, we summarize briefly the framework for plan recognition of (van Beek et al. 1993), which includes provision for initiating a clarification dialogue with the user.

The clarification process depends on two critical factors. First, the recognized plans are represented in a hierarchical plan library in the style of (Kautz 1987). For example, Figure 1 shows a graphical representation of a part of the plan library for a course-advising example[1]. The thick, gray arrows represent abstraction (or "isa") links and the thin, black arrows represent decomposition (or "subtask") links. Second, the plans are critiqued prior to clarification. Critiquing serves to label possible user plans with a fault annotation, from the following catalogue: (i) faultless (ii) failure of preconditions (iii) temporally inconsistent (iv) there exists a better plan.

Plan recognition can then be used to generate a response to a user in an advice-giving setting, augmenting "yes" or "no" answers with information about possible faults. There may be ambiguity about the user's plan, but if all possible plans have the same fault annotation or are faultless, then it is not necessary to resolve the ambiguity in order to formulate the response. In cases where the ambiguity matters, a clarification dialogue with the user is initiated.

The general algorithm for responding to a user question is as follows:

```
procedure GENERATE_RESPONSE(Query)
begin
    Check if Query is possible
    if Query fails then
        Output: "No, [Query] is not possible as [reason for
            failure]"
    else begin
        S <-- PLAN_RECOGNITION(Query)
        S <-- CRITIQUE(S)
        if AMBIGUITY_MATTERS(S) then
            S <-- CLARIFY(S)
        RESPOND(S)
          /* answer Query with a cooperative response */
    end
end
```

The clarification procedure traverses the plan library in a top-down order, maintaining a current branch point from which to select events (parts of possible plans) to ask the user. Initially, the current branch point is the *end* event, a distinguished node that is the root of the hierarchical plan library (see Figure 1). Based on the user's response, certain plans are eliminated from consideration. The process stops when ambiguity no longer matters to the final response. The clarification procedure makes use of a set S of possible plans, and is presented below:

```
procedure CLARIFY (S)
begin
    CB <-- end   /* set the current branch point */
    while AMBIGUITY-MATTERS(S) do
    begin
    if CB has only one child or one remaining event[2] then
        CB <-- next disjunctive branch point in top-down
                order
    select a disjunctive event (E) of CB to ask the user
    if user's answer is "yes" then
        S <-- (plans having E)
        CB <-- E
    else S <-- (S - (plans having E))
    end
    return (S)
end
```

Augmenting the Clarification Procedure

The algorithms of (van Beek et al. 1993) have some shortcomings. In particular, the criterion for selecting an event to ask about was left underspecified. We illustrated the possible use of likelihood factors, as a heuristic for selection, and mentioned the desirability of reducing the number of questions being asked overall.

We decided to examine more closely possible strategies for traversing the plan library in asking clarifying questions, and for selecting events to ask the user about. Our aim was to minimize the clarification dialogue, retaining coherence, in order to make the advice-giving session as agreeable as possible for the user.

The following capabilities have been introduced into our algorithms:

1. We have studied the options of continuing with "yes/no" questions from the system, as in (van Beek et al. 1993) or allowing menu-type selections from users. Our algorithms currently allow a general toggle into menu mode at the start of the system.

2. We have proposed a set of rules to determine the order in which the different alternatives of a current branch point are selected for clarification (in the case that menus are not used). We allow for likelihoods to be used in determining the selection. When these are impractical to introduce, we develop a default strategy, which determines the partitions of possible plans for a "no" reply to each alternative and then asks the user about the alternative which leaves the fewest remaining partitions. This is designed to eliminate as rapidly as possible, narrowing in on the plan of the user with a minimal number of questions asked.

3. We have studied the desirability of labeling certain events of the plan library as "key events", ones which will either substantiate or eliminate all plans of a fault partition. Key events can also be used as a selection criterion during clarification.

4. We have developed a strategy for avoiding pointless clarification, in the case that a user is intending a certain action which is inappropriate for an observation. An "overriding fault" is associated with such an action, and a response is provided which addresses the overriding fault.

---

[1] In this paper, we use the phrase "plan library" to refer to the subset of the entire domain library which has been isolated by the plan recognizer as the set of possible plans of the user.

[2] CB is used to determine which event to select; if CB has only one alternative, then there is no selection to be made and we update the branch point.

This strategy serves to reduce the length of the clarification dialogue.

5. We have developed a criterion for when it is possible to skip branch points in the event hierarchy, when traversing the library for clarifying events. Whenever each alternative of a current branch point is present in all remaining fault partitions, there is the possibility of skipping that branch point. We have also determined when we should skip branch points. Skipping branch points reduces the number of questions asked, and improves the former algorithm, while retaining the top-down order of traversal.

6. We have also developed a general algorithm, which can deal with complex hierarchical plans (ones where a conjunction of actions is necessary or multiple plans are being pursued). By maintaining the event hierarchy in a tree-like structure and placing conjunctive children onto a branch-point stack, we can clarify the alternatives of different sub-branches of complex plans in a depth-first manner. This more general algorithm is directly applicable to cases without complex plans (i.e. the same control and data structures may be used for the simpler cases as well).

In the sub-sections below, we summarize briefly each of the improvements described above, including examples for motivation.

The new basic clarification procedure

We first concentrated on specifying more precisely how to select an event to ask the user about, with our algorithm currently pointing to a current branch point (CB) of the plan library.

Van Beek et al. (1993) had assumed that "yes/no" questions would be asked, throughout the dialogue. With several alternatives at a current branch point, a selection must be made. Having a menu listing all possible selections avoids this issue. Yet, menus may be infeasible; for instance, in a dialogue over the telephone, it is unreasonable to expect a user to process a list of options. In fact, when menu lists become large, single questions may be preferable. If menus are not adopted, it is still possible to tie the selection of events to statistical information. This option was explored in the course advising implementation of (van Beek et al. 1993). If a system designer feels that these statistics are unreliable or unavailable, a separate selection criterion is required.

Our general clarify procedure accepts three options: CLARIFY_MENUS, CLARIFY_LIKELIHOOD, and CLARIFY_BASIC. The remaining discussion pertains to the CLARIFY_BASIC procedure. The GENERATE RESPONSE procedure is first augmented to track partitions of possible plans. After plans are critiqued, all plans with the same fault annotation are grouped into one partition. This results in **n** fault partitions. We then determine the partitions for a "yes" reply to each alternative of CB (termed a "yes" result). If every alternative does not have n partitions for its "yes" result, then we determine the "no" result for each alternative; i.e., the number of partitions that remain on a "no" reply to that alternative, and select to minimize dialogue as follows: a) select the event having the minimal "no" result; b) if there is a tie, select the event having the minimal "yes" result from among the tied events; c) otherwise, select randomly.

The rationale for selecting the event with the minimal "no" result is that if the reply is "yes", we are on the right track and if "no", we have eliminated the maximum number of partitions in comparison to any other alternative.

An example illustrates this heuristic most effectively. We have experimented with a course advising system, inspired by the actual division of departments at the University of Alberta. The example below employs a part of the plan library of Figure 1. Assume that the system has as a current branch point the event *Area of Science Concentration*. The user is asking a question about the course Calculus 102. His actual plan involves a specialization program in Computing Science, (but the system does not know this yet).

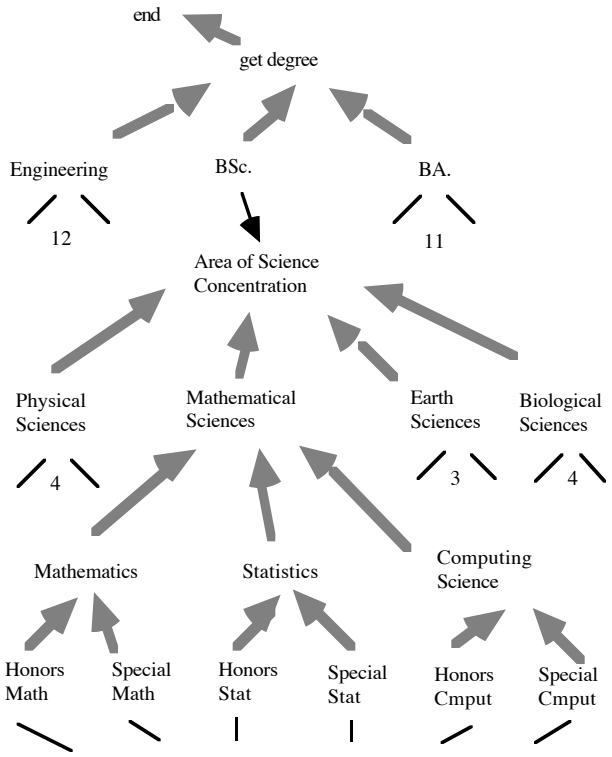

Figure 1 A portion of the hierarchy of the course-advisor There are 17 remaining plans (from our example library of 40). The plans were partitioned according to the fault annotations assigned by the plan critiquing phase, where all plans in a given partition are labeled with the same fault annotation. There are 8 remaining partitions for these plans, depicted below with the abbreviations (H) for honors and (S) for specialization. For example, all of the plans in partition p8 have the same single fault annotation of "there exists a better plan: Calculus 104" as it is strongly recommended that Mathematics and Computing Science students take Calculus 104 instead of Calculus 102, the course the user asked about.

p1: H-Genetics(BioSci)
p2: S-Genetics(BioSci)
p3: H-Microbiology(BioSci)
p4: H-Statistics(MathSci)
p5: S-Statistics(MathSci)
S-Microbiology(BioSci)
p6: H-Cartography(EarthSci)
S-Cartography(EarthSci)
p7: H-Chemistry(Physical)
S-Chemistry(Physical)
H-Physics(Physical)
S-Physics(Physical)
S-Geology(EarthSci)
p8: H-Mathematics(MathSci)
S-Mathematics(MathSci)
H-CompSci (MathSci)
S-CompSci(MathSci)

In this example, the CLARIFY_BASIC procedure is given the branch point event of *Area of Science Concentration*, which has four alternatives. With this particular situation, we have 8 remaining fault partitions (**n** = 8). In this case, the 'yes' result and 'no' result of each alternative is:

	Yes	No
Physical Sciences:	1	8 (a 'No' leaves 8 partitions)
Mathematical Sciences:	3	6
Earth Sciences:	2	7
Biological Sciences	4	5

Since all alternatives do not have a 'yes' result of **n** partitions (**n** = 8), we calculate the "no" results. We must now clarify the alternatives.

The minimum 'no' result is the one for the alternative of *Biological Sciences*. Asking this event will leave us with 5 remaining fault partitions on a 'No' reply, and 4 remaining fault partitions on a 'Yes' reply. According to our proposed rules, we would select the event of *Biological Sciences* to first ask the user (i.e., the minimal 'no' result). With this example, the user's reply would be 'No'. As a result of this reply, the possible plans are reduced to those plans which do not involve the event of *Biological Sciences*. When adjusting the plans, three partitions are eliminated, and one partition has a "Biological plan" removed. At this point the remaining partitions are as shown below.

p4: H-Statistics(MathSci)
p5: S-Statistics(MathSci)
p6: H-Cartography(EarthSci)
S-Cartography(EarthSci)
p7: H-Chemistry(Physical)
S-Chemistry(Physical)
H-Physics(Physical)
S-Physics(Physical)
S-Geology(EarthSci)
p8: H-Mathematics(MathSci)
S-Mathematics(MathSci)
H-CompSci (MathSci)
S-CompSci(MathSci)

In the algorithm, we determine the new set of possible plans. We are left with 5 partitions (**n** = 5). Since more than one alternative remains, another branch point has not yet been established and ambiguity still matters. Another alternative must be asked. The 'yes' result of the remaining alternatives does not change, but the 'no' results are now different. The results are:

	Yes	No
Physical Sciences:	1	5
Mathematical Sciences:	3	2
Earth Sciences:	2	4

Based on the new 'no' results, the next alternative to ask about is that of *Mathematical Sciences*. A 'No' reply to this event will leave two partitions (eliminating more partitions than any other alternative). In this case, the user's reply will be 'Yes'. The possible plans are now those plans which involve the event of *Mathematical Sciences*. After adjusting the plans, three partitions remain, as shown below.

p4: H-Statistics(MathSci)
p5: S-Statistics(MathSci)
p8: H-Mathematics(MathSci)
S-Mathematics(MathSci)
H-CompSci (MathSci)
S-CompSci(MathSci)

The CLARIFY_BASIC procedure returns the next branch point event of *Mathematical Sciences*, and we again need to clarify the alternatives of this event. The 'yes' and 'no' results for these alternatives are:

	Yes	No
Mathematics	1	3
Statistics	2	1
Computing Science	1	3

Using the selection rules, we first ask the user about entering the department of Statistics (rather than Mathematics or Computing Science). Since the user is planning to enter Computing Science, the reply is 'No'. After adjusting the plans, we are left with one partition (partition p8) with four remaining possible plans. Whenever we have only one partition, ambiguity no longer matters, so a response is provided. In this example, the response addresses the fault of the plans in partition p8: "The course Calculus 102 is permitted, but Calculus 104 is strongly recommended for students in Mathematics and Computing Science."

Focusing on key events during clarification

Once fault partitions have been identified, it is sometimes possible to identify key events, those which are shared by all the plans of a particular partition, and not shared by plans in other partitions. Selecting one of these events for clarifying questions may reduce the dialogue considerably. The following example illustrates the potential benefit of employing key events.

Suppose a user in the course advising domain has interacted with a menu-like interface and according to the user's profile, the 40 possible plans have been partitioned into two fault partitions. Partition-1 contains two faultless plans (honors and specialization Computing Science) and partition-2 contains the 38 other plans, each of which has the same fault annotation with respect to CMPUT-165. The basic approach would result in the following dialogue:

User: Should I enroll in CMPUT 165?
(The query results in 40 plans divided into 2 partitions.)
System: Do you intend to obtain
1. an Engineering degree
2. a Bachelor of Science degree

3. a Bachelor of Arts degree

User reply: 2.

(The reply eliminates Engineering and Arts plans, leaving 17 possible Bachelor of Science plans which are still divided into two partitions.)

System: Will you be concentrating in the area of

1. Physical Sciences
2. Mathematical Sciences
3. Earth Sciences
4. Biological Sciences

User reply: 2.

(The reply results in 6 remaining plans which all involve Mathematical Sciences, but we still have two fault partitions. Partition-1 still has two plans, but partition-2 now contains only 4 plans)

System: Do you intend to enter the department of

1. Mathematics
2. Statistics
3. Computing Science

User reply: 2.

(The reply eliminates all plans involving Mathematics and Computing Science, leaving only two Statistics plans. Partition-1 is now empty and is eliminated, leaving only partition-2, with the two possible plans. Since there is only one remaining partition, ambiguity no longer matters.)

System response: CMPUT 165 has a computing science prerequisite requirement and a computing science co-requisite requirement.

Exploiting the key event of *Computing Science*, which is shared by the two plans of partition-1, results in:

User: Should I enroll in CMPUT 165?

(The query results in 40 plans divided into 2 partitions.)

System: Do you intend to enter the department of Computing Science?

User reply: No.

(The reply eliminates all plans involving Computing Science. This eliminates partition-1, leaving only partition-2 with 38 possible plans. But ambiguity no longer matters.)

System response: CMPUT 165 has a computing science prerequisite requirement and a computing science co-requisite requirement.

If the user's reply were to be 'Yes', i.e., the user did intend to enter the department of Computing Science, rather than Statistics, the following dialogue would occur:

User: Should I enroll in CMPUT 165?

(The query results in 40 plans divided into 2 partitions.)

System: Do you intend to enter the department of Computing Science?

User reply: Yes.

(The reply eliminates all plans which do not involve Computing Science. This eliminates partition-2, leaving partition-1 with the remaining 2 possible plans. Ambiguity no longer matters.)

System response: CMPUT 165 is a good choice.

Thus, a single clarifying question is sufficient for providing a response.

Key events are not problem free to incorporate into the general clarification algorithms. When there are numerous fault partitions and thus numerous key events, there are cases where the overall coherence of the dialogue seems at risk, with a control structure which can now move freely to events within the plan hierarchy. Our current solution is to limit the use of key events to cases where there are three or less remaining fault partitions. Further details are provided in (Schmidt 1994).

Allowing overriding faults to avoid pointless clarification

The basic clarification algorithm will assume a top-down traversal of the plan library, from the top until ambiguity no longer matters. In some cases, critiquing may uncover a very serious fault which should be brought to the attention of the user; all further clarification questions are in some sense "pointless". We propose to terminate clarification in the presence of one of these "overriding faults", even though there is unresolved plan ambiguity which affects the precise response desired.

The following example illustrates overriding faults. In a cooking domain, the following exchange may occur:

User: Is it okay to make marinara sauce?
System: Are you making a meat dish?
User reply: Yes.
System: Do you intend to make a poultry dish?
User reply: No.
System: Do you intend to make a sea food dish?
User reply: No.
System: Regarding a beef dish, will you be serving steak?
User reply: Yes.
System: Do you intend to choose flank steak?
User reply: Yes.
(Assume that ambiguity no longer matters at this point)
System response: Yes, you can make marinara sauce, but your guest is a vegetarian, and you are out of flank steak, but the corner store has some.

In this dialogue, the additional information about flank steak might be appreciated, but it is pointless in light of the user's goal of entertaining a guest. A better dialogue is:

User: Is it okay to make marinara sauce?
System: Are you making a meat dish?
User reply: Yes.
(Ambiguity still matters, but the meat dish event should not be pursued)
System response: Yes, you can make marinara sauce, but your guest is a vegetarian.

The event "making a meat dish" has a serious fault so that the user should be warned directly.

The clarification procedure can be augmented to include overriding faults as follows. Overriding faults can be associated with the events that produce them during the critiquing process. Whenever we check to see if ambiguity still matters, we also check the event that is to be the next

branch point to see if it has an overriding fault, and if so, terminate with a response which attends to this fault.

Skipping branch points

Another setting where top-down traversal may be worth relaxing is one where a series of intervening questions will all eventually lead to a later question, necessary for the resolution of ambiguity. We characterize the set of plans in this case as one with a "diamond shape", where the paths in the hierarchy diverge from one event and later converge to another event (as illustrated in Figure 2). The problem is that not all diamond shapes allow questions to be skipped, without possibly missing the user's plan. We therefore had to develop a characterization of those settings in which diamond shaped plan sets should result in skipped branch points, during clarification.

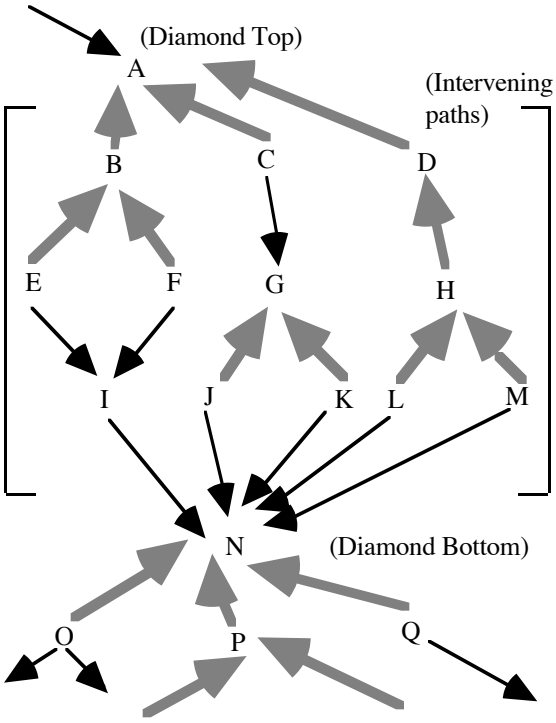

Figure 2 A diamond configuration

We only describe our solution briefly here. A situation in which each alternative of the current branch point has a "yes" result of **n** partitions and a "no" result of **n** partitions is one which may be skippable. We first determine a diamond bottom. We can only skip branch points to a diamond bottom if we can resolve the ambiguity without resorting to intervening events. That is, we determine if we can reduce the partitions to one partition by only asking those events which occur below the diamond bottom. As a result of this characterization, the basic clarification algorithm can be augmented to check for skippable settings, resulting in further reductions to the clarification dialogue. See (Schmidt 1994) for a full algorithm and further discussion.

Allowing complex plans

Another direction for improving our clarification dialogue algorithms is to extend the coverage to process complex plans (ones where conjuncts of events are necessary). For example, we might allow multiple events to be specified by users. In a cooking advice domain, a user may ask: "Is it okay to serve French wine with a marinara sauce?". The possible plan of the user may still be ambiguous (is he planning pasta or meat?) and further clarification may be important (perhaps a red wine or a white wine is a restriction). In cases such as these, the challenge in designing a clarification procedure is to track multiple branch points in a plan hierarchy, deciding which branch to pursue first, in order to minimize the length of the dialogue.

We have developed an extended algorithm, which allows complex events. Moreover, all the improvements introduced for the single event case can be easily incorporated into this more general framework (e.g. can check for key events in fault partitions). The solution requires a new data structure, the branch point stack, to retain conjuncts of plans. We omit the details of this part of our work, but see (Schmidt 1994) for further elaboration.

Summary

The algorithms of (van Beek et al. 1993) for generating clarification dialogue were modified, to allow the dialogues to be reduced. Data structures for fault partitions and a stack of branch points were introduced. A default criterion for selecting events was outlined, based on partitions. Cases for circumventing the top down traversal of the plan library, to focus on key events, terminate the dialogue prematurely or skip insignificant parts of the dialogue were all identified and incorporated into the overall procedure. Complex events could now be handled and design options of employing menus or likelihoods easily accommodated.

Discussion

In this paper, we have presented a framework for soliciting clarification from users during plan recognition.

We have proposed several improvements to earlier algorithms. These improvements have been made possible primarily through the specification of a new data structure, the fault partition, and a thorough investigation of how fault partitions may be exploited to select clarifying events.

Our system has been designed to prefer a top-down traversal of the set of possible plans. One reason why this makes sense is to improve dialogue coherence. In our testing of the course advising and cooking domains, we discovered examples where the hierarchy was quite "bushy". These cases would result in either very long menus of questionable coherence or lengthy "yes/no" dialogues, if control were bottom-up. The ability to prune from generalization to specialization in the top-down order was clearly defined and useful for extending to the case of complex plans.

In general, it is important to consider the overall coherence of a clarification dialogue. At present, there are few theoretical measures to evaluate how understandable a session will be for a user. We have encoded a subset of the actual course advising domain of the University of Alberta. As a next step, we can approach sample users and then perhaps an empirical evaluation of coherence could be conducted.

Calistri-Yeh (1991) has also advocated the introduction of user models to handle plan ambiguity. He introduces probabilities in order to determine the user's plan. Although this solution is effectively user-specific (like ours), it admits incorrect selection during plan recognition, leading to a debugging dialogue with the user. Our approach is to reduce possibilities by engaging in clarification, never overcommitting. Likelihoods are used to assist in selecting questions for the user, but it is the user's answers to these questions which determines whether to prune away a possible plan.

Wu (1991) has proposed engaging in clarification with users, with his approach of "actively acquiring" knowledge from the user in various circumstances, including cases of plan ambiguity and novel plan recognition. The focus of this work is somewhat different, allowing plan recognition to fail entirely. The paper addresses when clarifying questions are necessary to introduce. There is, however, no independent proposal for selecting effective clarifying questions or reducing the length of the clarification dialogue.

Nonetheless, the topic of allowing novel plans to be recognized is important. Our implementation has built on Kautz's style of plan recognition. Goodman and Litman (1992) criticize this plan recognition system for assuming a complete plan library. For future work, we can study how to extend our clarification dialogue algorithm to allow users with novel plans (ones not initially recorded in the plan library). This would be valuable for improving the interface of a system designed to allow updates to the plan library during plan recognition, such as that of (Cohen and Spencer 1993).

In fact, effective design of a plan library is critical for producing coherent clarification dialogues of reasonable length. For example, a hierarchy with very few specialization levels may lead to an impoverished "context" for user comprehension (e.g. "Are you planning to go to university?", "Are you considering the math with minor in physics option?"). Developing guidelines for the designers of plan libraries would be another valuable research topic.

Our framework for generating clarifying questions identifies the nodes of a plan hierarchy which should be used for each question. We have not studied deeper natural language generation issues, such as selecting the appropriate linguistic form of the question or of the final response. This is another avenue for future work. As a starting point, we could look at systems for natural language generation such as that of McKeown (1985) or text planners such as that of Moore and Paris (1993).

In summary, we have gained important insights into how to engage users in clarification during plan recognition. With our expanded algorithms and a practical domain in hand we are ready to experiment to uncover new directions and improvements.

References

Allen, J.; A plan-based approach to speech act recognition; Technical Report 131, Department of Computer Science, University of Toronto, 1979.

Calistri-Yeh, R.; Utilizing user models to handle ambiguity and misconceptions in robust plan recognition; *User Modeling and User-Adapted Interaction*, **1**:289-322, 1991.

Carberry, S.; Tracking user goals in information-seeking environment; In Proceedings of the National Conference on Artificial Intelligence, pages 59-63, 1983.

Cohen, R., Spencer, B. and van Beek, P.; In search of practical specifications - allowing the user a more active role; AAAI Spring Symposium on Implemented KR and Reasoning Systems, March 1991.

Cohen, R. and Spencer, B.; Specifying and updating plan libraries for plan recognition; Proceedings of 9th IEEE conference on AI applications (CAIA 93), March 1993.

Goodman, B. and Litman D.; On the interaction between plan recognition and intelligent interfaces; *User Modeling and User-Adapted Interaction* , **2:**83-115, 1992.

Kautz, H.; *A Formal Theory of Plan Recognition.* ; PhD thesis, University of Rochester, 1987. Available as: Department of Computer Science Technical Report 215.

McKeown, K.; Text generation: using discourse strategies and focus constraints to generate natural language text; Cambridge University Press, 1985.

Moore, J. and Paris, C.; Planning text for advisory dialogues: capturing intentional and rhetorical information; *Computational Linguistics*, **19**:651-694, 1993.

Schmidt K.; Clarification dialogues for plan recognition in advice-giving settings; M.Sc. thesis, Dept. of Computing Science, University of Alberta, April 1994.

van Beek, P.; A model for generating better explanations; In Proceedings of the 25th Conference of the Association for Computational Linguistics, pages 215-220, 1987.

van Beek, P. and Cohen, R.; Towards user-specific explanations from expert systems; In Proc. of the Sixth Canadian Conference on Artificial Intelligence, pages 194-198, 1986.

van Beek, P., Cohen, R., and Schmidt, K.; From plan critiquing to clarification dialogue for cooperative response generation; *Computational Intelligence*, **9:**132-154, 1993. An earlier version appears in IJCAI-91, pages 938-944.

Wu, D.; Active acquisition of user models: implications for decision-theoretic dialog planning and plan recognition; *User Modeling and User-Adapted Interaction*, **1**:149-172, 1991.