Resolving Plan Ambiguity for Response Generation

Peter van Beek and Robin Cohen
Department of Computer Science
University of Waterloo
Waterloo, Ontario
CANADA N2L 3G1

Abstract

Recognizing the plan underlying a query aids in the generation of an appropriate response. In this paper, we address the problem of how to generate cooperative responses when the user's plan is ambiguous. We show that it is not always necessary to resolve the ambiguity, and provide a procedure that estimates whether the ambiguity matters to the task of formulating a response. If the ambiguity does matter, we propose to resolve the ambiguity by entering into a clarification dialogue with the user and provide a procedure that performs this task. Together, these procedures allow a question-answering system to take advantage of the interactive and collaborative nature of dialogue in recognizing plans and resolving ambiguity.

Introduction

Somewhat obviously, plan recognition is the process of inferring an agent's plan from observation of the agent's actions. The agent's actions can be physical actions or speech actions. Four principal methods for plan recognition have been proposed in the literature. The methods are plausible inference (Allen [1], Carberry [2], Litman [15], Sidner [25]), parsing (Huff and Lesser [9]), circumscribing a hierarchical representation of plans and using deduction (Kautz [12, 13]), and abduction (Charniak and McDermott [6], Konolige and Pollack [14], Poole [24]).

Our particular interest is in the use of plan recognition in question-answering systems, where recognizing the plan underlying a user's queries aids in the generation of an appropriate response. Here, the plans and goals of the user, once recognized, have been used to: supply more information than is explicitly requested (Allen [1], Luria [16]), handle pragmatically ill-formed queries and resolve some intersentential ellipses (Carberry [2, 3, 4]), provide an explanation from the appropriate perspective (McKeown et al. [17]), respond to queries that result from an invalid plan (Pollack [20, 21, 22]), and avoid misleading responses and produce user-specific cooperative responses (Joshi et al. [10, 11], van Beek and Cohen [26, 27], Cohen et al. [7]).

Example 1 (Joshi et al. [11]). As an example of a cooperative response consider the following exchange between student and student-advisor system. The plan of the student is to avoid failing the course by dropping it.

User: Can I drop numerical analysis?
System: Yes, however you will still fail the course since your mark will be recorded as withdrawal while failing.

If the system just gives the direct answer, "Yes" the student will remain unaware that the plan is faulty. The more cooperative answer warns the student.

An important weakness of this work in response generation, however, is the reliance on a plan recognition component being able to uniquely determine the plan of the user. This is clearly too strong an assumption as the user's actions often will be consistent with more than one plan, especially after only one or a few utterances when there is insufficient context to help decide the plan of the user. In Example 1 there are many reasons why a student may want to drop a course, such as resolving a scheduling conflict, avoiding failing the course, or finding the material uninteresting. There may be no reason to prefer one alternative over the other, yet we may still want to generate a response that does more than just give a direct answer to the user's query.

In this paper, we address the problem of what the system should do when the user's actions are ambiguous as they are consistent with more than one plan. To the extent that this problem has been considered by researchers in plan recognition, it is generally assumed that the plan recognition system overcomes the ambiguity problem by inferring the most likely interpretation, given its assessment of the context and dialogue so far and knowledge of typical plans of action. Thus there is a dependence on salience heuristics to solve the ambiguity problem [e.g. 1, 2, 17, and see the final section]. Existing proposals for resolving ambiguity beyond heuristics are underspecified and what usually underlies these proposals is the assumption that we always want to determine one unique plan [2, 15, 19, 25].

We show how to relax the assumption that the plan recognition component returns a single plan. That is, given that the result of the plan recognition phase will usually be a disjunction of possible plans, we show how to design a response component to generate cooperative responses given the disjunction. We show that it is not always necessary to resolve ambiguity, and provide a procedure that allows the

response component to estimate whether the ambiguity matters to the task of formulating a response. If the ambiguity does not matter, the response component can continue to answer the user's queries and ignore the ambiguity in the underlying goals and plan until further queries help clarify which plan the user is pursuing. If the ambiguity does matter, the system should take advantage of the interactive and collaborative nature of dialogue in recognizing plans and resolving ambiguity. A key contribution of this work therefore is providing a clear criterion for *when* to respond to a question with a question that will differentiate between some of the possibilities. We also propose a specific solution to *what* questions should then be asked of the user. Moreover, these questions are asked only to resolve the ambiguity to the point where it no longer matters (this is not necessarily to a unique plan).

Example 2. Here are two different examples to give a flavor of what we are proposing. There are two agents: a cook and an expert who is cooperative, helpful, and adept at recognizing plans. The expert observes the cook making marinara sauce and recognizes the cook could be pursuing three possible plans, all with the same top level goal of preparing a meal: make fettucini marinara or spaghetti marinara (both a pasta dish) or chicken marinara (a meat dish).

a. Suppose the cook then asks the expert: "Is a red wine a good choice?" The expert has the criteria for wine selection that red wine should be served if the meal is chicken, fettucini marinara, or spaghetti marinara and white if fettucini alfredo. There is enough information for the expert to decide that red wine should be bought and the ambiguity does not need to be resolved to cooperatively answer the question.

b. Now suppose the expert also knows that the guest of honor is allergic to gluten and so would not be able to eat if a pasta dish was served. Here the ambiguity is important as the expert has recognized that the cook's prepare-a-meal goal may conflict with the cook's entertain-important-guest goal. The expert will want to resolve the ambiguity enough to be assured that the proposed meal does not include a pasta dish and so clarifies this with the cook.

Estimating Whether the Ambiguity Matters

Example 2, above, showed that sometimes it is necessary to resolve ambiguity and sometimes it is not. Here we give criteria for judging which is the case. The result is a procedure that allows the response component to estimate whether the ambiguity matters to the task of formulating a response.

Assuming we can answer the user's query, deciding when we want to give more than just a direct answer to the user's query depends on the plan of the user. Previous work has identified several kinds of faults in a plan that a cooperative response should warn a user about. We generalize this work in identifying faults and call it *plan critiquing*. Our proposal therefore is to first apply a plan critiquing procedure to determine possible faults.

Plan Critiquing

A plan may be labeled as faulty if it will fail to achieve its high level goal (e.g. Allen [1]) or if there are simply better alternatives for reaching that goal (e.g. Pollack [20]). Joshi et al. [10, 11] formalize the above and identify additional cases (such as warning the user that a certain plan is the only way to achieve a goal).

In [26, 27], we make some extensions to Joshi et al. [10, 11] and give a procedure to determine faults in plans and to address these faults through cooperative responses. Faults include both plans which fail and plans which can be replaced by better alternatives. In [26, 27], we also include the critical extension of domain goals. To explain, the system needs to not only respond to goals inferred from the current discourse but also to the domain goals a user is likely to have or known to have even though they are not stated in, or derivable from, the discourse.

Example 3.

User: I'm not interested in the material and so would like to drop the course. Can I?

The ideal response should say more than just "Yes", but also warn the student that the domain goal of achieving a degree conflicts with the immediate goal of dropping the course. This example shows how *competing* goals may exist and need to be addressed in a response.

To determine whether the ambiguity matters, we propose to apply an extension of the procedure of [26, 27], which will be sensitive to complementary goals as well. The standard assumption in cooperative response work is that the user is pursuing only a single chain of goals; we allow actions to be used to achieve more than one *complementary* goal. For example, in the course-advising domain we can assume that all users have the goal to avoid failing a course. If a user then asks "I'm not interested in the course and so would like to drop it. Can I?" the response should address not just the goal of avoiding uninteresting courses but also the complementary goal of avoiding failing courses. For example, the response should warn the user if he will fail the course because of withdrawal while failing (as in Example 1).

The algorithm to estimate whether the ambiguity matters is below. The deciding criterion is whether the critiques for all plans are the same. By same critique or same fault (Case 1. b. in the algorithm) we mean, for example, same better way or same conflict with competing domain goal.

Input: A set of possible plans (the output from a plan recognition algorithm).
Output: The set of possible plans with critiques attached to each plan.

Algorithm *Critique:*
begin
for each plan in the set of possible plans **do**
critique the plan and, if there is a fault, annotate the plan with the fault

Input: A set of critiqued possible plans.
Output: "Yes", the ambiguity matters, or "No" the ambiguity does not matter.

Algorithm *Ambiguity_matters:*

We are in one of the following two cases:

Case 1. "No", the ambiguity does not matter.
The critiques are the same for all the plans. That is, either
a. every plan is faultless, or
b. every plan is annotated with the same fault.

Case 2. "Yes", the ambiguity does matter.
The critiques are different for some or all of the plans. That is, either
a. some, but not all, of the plans are faultless (the faults may or may not be the same), or
b. every plan is annotated with a fault and the faults are not all the same.

end

An Example to Illustrate the Proposal

Suppose the user asks "Can I drop numerical analysis?". First, a plan recognition algorithm is called to determine the possible plans of the user. They are found to be:

end ⇐ resolve schedule conflict → drop course
end ⇐ avoid uninteresting prof → drop course

Second, algorithm *Critique* is called to critique the plans. As a result, both plans are labeled with the same fault that there is a better plan for achieving the goal. Third, algorithm *Ambiguity_matters* is called to determine whether the ambiguity regarding the plan of the user matters to the task of formulating a response. It is found that the ambiguity does not matter as both plans are annotated with the same fault (Case 1.b of the algorithm). Finally, the critiqued plans are passed to a response generation procedure. The answer then given in this example is, "Yes, but a better way is to switch to another section."

In general, in (Case 1.a) a response generation procedure can just give a direct answer to the user's query, and in (Case 1.b) can give a direct answer plus any warranted additional information, such as telling the user about the fault.

In the above example it was found that the ambiguity did not matter as there was enough information to generate a cooperative response. If instead it was found that the ambiguity did matter (Case 2 of the algorithm) we propose that we we enter into a clarification dialogue with the user to resolve the ambiguity to the point where it no longer does matter. That is, until we are in Case 1. A response generation procedure would then be called.

Clarification Dialogues: Questions in Response to Questions

What should we ask the user when a clarification is necessary? Clearly, we do not want to simply list the set of possible plans and ask which is being pursued. Below is an algorithm that determines what to say. Our algorithm for estimating whether the ambiguity matters is not dependent on the method of plan recognition used. Now our proposal for clarification dialogues is tied to a hierarchical plan library in the style of Kautz [12]. The input to the algorithm is a set of possible plans where the user's action is related to top-level or end goals through chains of goals. Each plan in the set is annotated with a a critique. The key idea is to ask about the highest level possible, check whether the ambiguity still needs to be further resolved, and if so, ask at the next level down, iteratively, through the hierarchy of goals.

Input: A set of critiqued possible plans (the output from algorithm *Critique*).
Output: The pruned set of possible plans such that the ambiguity no longer matters.

Algorithm *Clarify:*
begin
initialize the *current level* to be the first branch point from the top in the set of possible plans

while *Ambiguity_matters* = "Yes" **do**

separate out the distinct goals in the set of remaining possible plans that are one level below the *current level* and are annotated with a fault

list the goals (perhaps with their accompanying annotations as justification for why we are asking) and ask the user whether one of them is part of the plan being pursued

according to the answer, remove the plans that are not being pursued from the set of possible plans and update the current level in the hierarchy that is being looked at to be the next branch point

end while
end

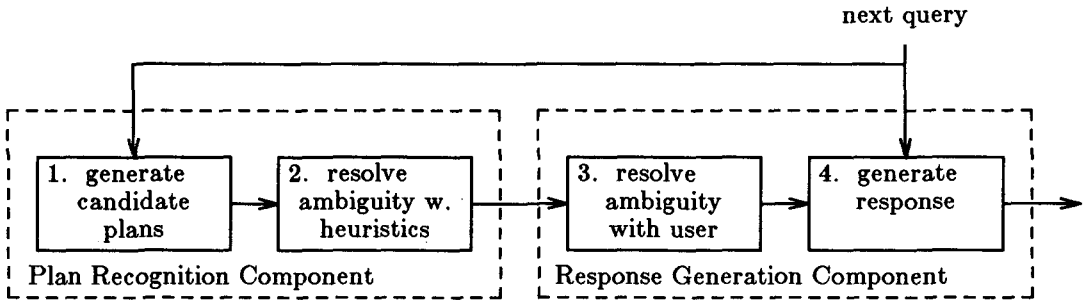

Fig. 1. Major modules of our proposed query-answering system

Example 2b (Revisited). In our story in the introduction about cooking for our allergic guest the expert has recognized the following three plans:

1. end ⇐ prepare meal ⇐ make meat dish ⇐ make chicken marinara → make marinara
2. end ⇐ prepare meal ⇐ make pasta dish ⇐ make fettucini marinara → make marinara
3. end ⇐ prepare meal ⇐ make pasta dish ⇐ make spaghetti marinara → make marinara

Using the algorithm of the previous section, the three plans are critiqued and it is found that the ambiguity matters. The plan involving a meat dish is found to be faultless but the two plans involving a pasta dish are found to conflict with another goal of the cook: to entertain the guest. Using the algorithm of this section, the question asked to resolve the ambiguity would be "Are you making a pasta dish (perhaps with justification of why we are asking)?" After either answer of yes or no we know enough that the ambiguity no longer matters. Note that if we just ask the more general "What are you making?" this allows such uninformative responses as "dinner" or just "you'll see".

When asking a question we propose to ask about as high a level of goal as possible that still helps to resolve the ambiguity and to work top down. A top down approach is better as it provides a useful context for any later queries and makes users give us an answer at a higher level than they may otherwise do. Moreover, the user may be mistaken about decompositions or have some other wrong view of the domain and by stepping downward through the paths of possible plans these misconceptions may be detected. Here is an example. Suppose the user asks "Can I take numerical analysis?". The system recognizes two plans.

end ⇐ get_degree ⇐ B.A. ⇐ electives → course
end ⇐ get_degree ⇐ B.Sc. ⇐ required → course

a. System: Are you pursuing a B.Sc.?
b. System: Are you trying to fulfill your elected or required courses?

Question (a) is what our algorithm would ask. Question (b) is what a procedure which uses a bottom up approach would ask. But (b) allows potential confusion to go undetected. The user could answer "required", believing that the course is required for a B.A., for example. Thus, we advocate Kautz style plan recognition [12], as other plan recognition methods [2, 15, 25] would stop after branch points points and thus could only propose electives and required as the two possible plans. Question (b) is the only question this previous work could ask the user.

Starting with the top most goals and working down may sometimes give as many questions as bottom up approaches. However, others have noted that bottom up dialogues are difficult to assimilate without misinterpretation [2, p. 54]. Therefore, we maintain that the top down approach is more desirable. Moreover, some higher level questions, such as question (a), above, can be eliminated using goals known from the previous discourse or background knowledge about the user.

Current extensions we are examining include allowing the user to have multiple goals so that more than one path from top level goals to action may be correct. This requires resolving ambiguity in multiple paths through the set of possible plans. This could be done in a depth-first manner, using clue or redirection words to guide the user when we return to resolve the ambiguity in the other branches. We are also investigating selecting the minimal subset of the possible plans from those with faults and those without faults (at the moment the algorithm always takes the subset with faults).

Discussion and Conclusion

In this section we summarize our proposals and defend our position that this straightforward way of doing things is a good way. With reference to Fig. 1, we discuss the design of boxes 2, 3, and 4 and the tradeoffs involved between boxes 2 and 3.

Box 2: Resolve the ambiguity with heuristics. As mentioned earlier, many researchers have proposed heuristics to prefer one plan over another [1, 2, 17, 8, 18, 12, 23, 14]. Some of these heuristics can be incompatible with cooperative response

generation. For example, Allen's [1] preference heuristics are generally incompatible with recognizing and responding to faulty plans (such as the response in Example 1). Because we are using plan recognition for response generation, this should affect the design of Box 2 and therefore what gets passed to Box 3.

Box 3: Resolve the ambiguity with the user. Previous work in response generation makes the assumption that what gets passed to the RG component is a single plan the PR component proposes the user is pursuing. We argue that, unless we are willing to sometimes arbitrarily commit to one plan instead of another, there will be times when one plan cannot be chosen over another and therefore there will be ambiguity about which plan the user is pursuing. Result: we need a method to resolve the ambiguity. In plan recognition in a discourse setting (as opposed to key-hole recognition), the goals and plan the user holds are knowable simply by asking the user. But we do not want to always just ask if it is not necessary so we need to know when to start a clarification dialogue and what to say. And when we do ask, we want to ask the minimal number of questions necessary to resolve the ambiguity until it no longer matters. To this end, box 3 contains a procedure that estimates by plan critiquing whether the ambiguity matters to the task of formulating a response. If the ambiguity does not matter the result is passed to box 4. If the ambiguity does matter, a procedure is called that starts a clarification dialogue, responding to the user's question with questions that iteratively differentiate between the possibilities.

Box 2 vs. Box 3: The tradeoffs. Much previous work in plan recognition makes the assumption that we want the PR component to commit to and return a single plan. Carberry and McKeown, for example, use a strong heuristic to commit to a single plan [2, 17]. However, this means the system will at times commit to the wrong plan. Doing it this way requires the ability to handle natural language debugging dialogues. Why we do not want to commit to a single plan and then, if we are wrong, repair using a debugging dialogue? Carberry [5, p.4] argues that a system will appear "unintelligent, obtuse, and uncooperative" if it engages in lengthy clarification dialogues. However, a procedure to perform a debugging dialogue is not specified and is, we speculate, a difficult problem. We argue for not committing early. Our hypothesis is that a clarification dialogue is better than a debugging dialogue. The questions in the clarification dialogues are simple to answer, whereas determining that the system has misunderstood your goals and plan requires users to engage in plan recognition. That is, users must recognize the plan the RG component is using from its responses and note that it differs from their plans. Moreover, the user may not recognize we are wrong and be mislead. Finally, we argue that, if the questions are carefully chosen, the clarification dialogues need not be lengthy or too frequent. Note that preference heuristics can still be used in our approach. These would best be applied when too many top level goals give an unwieldy clarification question.

Box 4: Generate the response. Once Box 3 has estimated that any remaining ambiguity does not matter to generating a cooperative response, the disjunction of possible plans is passed to Box 4. There are two cases; both can now be handled as in previous work except that there is now the additional complication that we allow one action to contribute to more than one chain of goals. The response component must then generate a conjunctive response that addresses each of the goals.

1. Every plan is faultless, so we just give a direct answer to the user's query but ignore the underlying goals and plan until further queries help clarify which plan the user is pursuing, and
2. Every plan has the same fault, so we give a direct answer plus some additional information that warns the user about the deficiency and perhaps suggests some alternatives (see [10], [26]).

Soap Box: This paper offers an important contribution to natural language generation. It discusses a clear criterion for *when* to initiate a clarification dialogue and proposes a specific solution to *what* questions should be asked of the user to achieve clarification. We believe that natural language response generation systems should be designed to involve the user more directly and this is sometimes achieved quite successfully with our proposals.

There may be tradeoffs between overcommitting in the plan recognition process and engaging in lengthy clarification dialogue, particularly with a large set of complex candidate plans. This may suggest applying pruning heuristics more actively in the plan recognition process (Box 2) to reduce the number of questions asked in the clarification dialogue (Box 3). For future work, these tradeoffs will be examined more closely as we test the algorithms more extensively.

Acknowledgements. We thank Fei Song and Bruce Spencer for comments on an earlier version of this paper and for many discussions about plan recognition. Financial assistance was received from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] Allen, J. F. 1983. Recognizing Intentions from Natural Language Utterances. In *Computational Models of Discourse,* M. Brady and R. C. Berwick (eds.). MIT Press.

[2] Carberry, S. 1985. Pragmatic Modeling in Information System Interfaces. Ph.D. thesis available as University of Delaware Technical

Report 86-07, Newark, Del.

[3] Carberry, S. 1988. Modeling the User's Plans and Goals. *Computational Linguistics* **14**, 23-27.

[4] Carberry, S. 1989. A Pragmatics-Based Approach to Ellipsis Resolution. *Computational Linguistics* **15**, 75-96.

[5] Carberry, S. 1989. A New Look at Plan Recognition in Natural Language Dialogue. University of Delaware Technical Report 90-08, Newark, Del.

[6] Charniak, E., and D. V. McDermott. 1985. *Introduction to Artificial Intelligence.* Addison Wesley.

[7] Cohen, R., M. Jones, A. Sanmugasunderam, B. Spencer, and L. Dent. 1989. Providing Responses Specific to a User's Goals and Background. *International Journal of Expert Systems: Research and Applications* **2**, 135-162.

[8] Goldman, R., and E. Charniak. 1988. A Probabilistic Assumption-Based Truth Maintenance System for Plan Recognition. *Proc. of the AAAI-88 Workshop on Plan Recognition,* St. Paul, Minn.

[9] Huff, K., and V. R. Lesser. 1982. Knowledge-Based Command Understanding: An Example for the Software Development Environment. Computer and Information Sciences Technical Report 82-6, University of Massachusetts at Amherst, Amherst, Mass. Cited in [13].

[10] Joshi, A., B. Webber, and R. Weischedel. 1984. Living up to Expectations: Computing Expert Responses. *Proc. of the Third National Conference on Artificial Intelligence,* Austin, Tex., 169-175.

[11] Joshi, A., B. Webber, and R. Weischedel. 1984. Preventing False Inferences. *Proc. of 10th International Conference on Computational Linguistics (COLING),* Stanford, Calif., 134-138.

[12] Kautz, H. A. 1987. A Formal Theory of Plan Recognition. Ph.D. thesis available as University of Rochester Technical Report 215, Rochester, N.Y.

[13] Kautz, H. A., and J. F. Allen. 1986. Generalized Plan Recognition. *Proc. of the Fifth National Conference on Artificial Intelligence,* Phil., Penn., 32-37.

[14] Konolige, K., and M. E. Pollack. 1989. Ascribing Plans to Agents. *Proc. of the Eleventh International Joint Conference on Artificial Intelligence,* Detroit, Mich., 924-930.

[15] Litman, D. J., and J. F. Allen. 1987. A Plan Recognition Model for Subdialogue in Conversations. *Cognitive Science* **11**, 163-200.

[16] Luria, M. 1987. Expressing Concern. *Proc. of the 25th Conference of the Association for Computational Linguistics,* Stanford, Calif., 221-227.

[17] McKeown, K. R., M. Wish, and K. Matthews. 1985. Tailoring Explanations for the User. *Proc. of the Ninth International Joint Conference on Artificial Intelligence,* Los Angeles, Calif., 794-798.

[18] Neufeld, E. 1989. Defaults and Probabilities; Extensions and Coherence. *Proc. of the First International Conference on Principles of Knowledge Representation and Reasoning,* Toronto, Ont., 312-323.

[19] Perrault, C. R., J. F. Allen, and P. R. Cohen. 1978. Speech Acts as a Basis for Understanding Dialogue Coherence. *Proc. of the 2nd Conference on Theoretical Issues in Natural Language Processing (TINLAP),* Urbana-Champaign, Ill., 125-132.

[20] Pollack, M. E. 1984. Good Answers to Bad Questions: Goal Inference in Expert Advice-Giving. *Proc. of the Fifth Canadian Conference on Artificial Intelligence (CSCSI/SCEIO),* London, Ont.

[21] Pollack, M. E. 1986. Inferring Domain Plans in Question-Answering. Ph.D. thesis (Univ. of Penn.) available as SRI International Technical Note 403, Menlo Park, Calif.

[22] Pollack, M. E. 1986. A Model of Plan Inference that Distinguishes Between the Beliefs of Actors and Observers. *Proc. of the 24th Conference of the Association for Computational Linguistics,* New York, N.Y., 207-214.

[23] Poole, D. L. 1985. On the Comparison of Theories: Preferring the Most Specific Explanation. *Proc. of the Ninth International Joint Conference on Artificial Intelligence,* Los Angeles, Calif., 144-147.

[24] Poole, D. L. 1989. Explanation and Prediction: An Architecture for Default and Abductive Reasoning. *Computational Intelligence* **5**, 97-110.

[25] Sidner, C. L. 1985. Plan Parsing for Intended Response Recognition in Discourse. *Computational Intelligence* **1**, 1-10.

[26] van Beek, P. 1987. A Model For Generating Better Explanations. *Proc. of the 25th Conference of the Association for Computational Linguistics,* Stanford, Calif., 215-220.

[27] van Beek, P., and R. Cohen. 1986. Towards User-Specific Explanations from Expert Systems. *Proc. of the Sixth Canadian Conference on Artificial Intelligence (CSCSI/SCEIO),* Montreal, Que., 194-198.