Highlighting User Related Advice

Kathleen R. McKeown and Robert A. Weida
Department of Computer Science
450 Computer Science Building
Columbia University
New York, N.Y. 10027
CUCS-345-88

ABSTRACT: Research on explanation techniques for expert systems has demonstrated that (1) explanations are most effective when they address the user's needs and (2) it is necessary to augment explanations with information that is missing from the expert system's reasoning. It is our thesis that explanation content can also be improved by removing extraneous information from the system's reasoning and reorganizing the remainder to emphasize user concerns. To test our ideas, we have developed an interactive natural language problem-solving system called ADVISOR which advises students on course selection. Previously, we have reported on our methodology for deriving user goals from the discourse, representing different points of view in the knowledge base and inferring user-oriented advice with a rule-based system that employs information from the appropriate perspective to address user goals. In this paper, we describe a model for pruning an explanation to highlight the role of the user's goal. The model is part of ADVISOR's natural language generation component. We demonstrate its efficacy with examples of different advice that ADVISOR provides for the same query in the context of different goals.

WORD COUNT: 3487 (3118 in text, 377 in display). In addition, Figure 1 contains a diagram of approximately 1/2 page.

TOPIC: Natural Language

1 Introduction

Research on explanation (e.g., [6, 9, 3]) has shown how an underlying knowledge base must be abstracted and augmented to provide useful explanations. In our previous work, we have shown that advice can be more effective if it addresses the advisee's goals and have also used an augmented rule base and supporting knowledge base as part of our system to produce goal oriented explanations [7, 8]. While the aim of augmenting the knowledge base is to recover and abstract information *missing* in previous explanation systems, current explanations can also be improved by removing *extraneous* information: in particular, information that does not address a user's needs. If goal related advice is buried amid other information needed to produce a response, its effectiveness is diminished. The user must be able to easily identify how the system has taken his/her needs into account to be convinced that the advice provided is worth taking.

In this paper, we present a model for highlighting goal related advice in an explanation by reorganizing and pruning a tree-structured inference trace produced by a rule-based expert system. Our model is part of a natural language surface generator that produces the actual text of an explanation. The generator traverses the inference trace, potentially producing a phrase for each node of the trace. It prunes subtrees from the trace that do not show how the user's goal is advanced, depending on circumstances. When choosing vocabulary to realize individual rules, or nodes, of the trace as English phrases, the natural language generator makes use of tree context, conversational principles, and semantic implications of selected words to reorder and remove unnecessary information. The model has been implemented as part of a student advisor system, called ADVISOR.

In earlier work, we showed how to derive domain goals from a discourse segment and tie them to explanation content as part of ADVISOR, although the system didn't generate explanations [7]. In more recent work, [8], we presented an overview of

techniques ADVISOR now uses to tailor explanations. The overview included descriptions of how ADVISOR uses a partitioning of the knowledge base into perspectives to determine information that is potentially relevant to a user's goal, expert system rules that specify how actions may advance users' goals, and one type of pruning. In contrast, this paper provides details on ADVISOR's full model for pruning and reorganizing content.

2 System Overview

Our method has been implemented as part of an ongoing project to develop a dialogue facility for computer-aided problem solving. ADVISOR can provide information about courses and advice about whether a student can or should take a particular course. The system is structured as a question-answering system which invokes an underlying expert system on receiving "can" questions (e.g., "Can I take natural language this semester?") and "should" questions (e.g., "Should I take data structures?"). This production system uses its rule base to determine the advice provided (i.e., *yes* or *no*) and the trace of rule invocations is used to provide a supporting explanation of the advice.

Figure 1 shows a diagram of ADVISOR's modules and their interaction. ADVISOR uses an ATN parser [13] and Woods-style semantics [12] to produce a literal interpretation of a user's question. A goal inferencer using an extended version of Allen and Perrault's [1] inferencing method derives the domain goal for both the current question and the discourse so far. It makes use of a database of domain plans to do so. Static information about courses (e.g., when they are offered, teachers, topics covered, etc.) is stored in a KL-ONE style knowledge base [2]. This is used to answer information type questions (e.g., "Who teaches natural language?") and to provide information needed for inferencing. An expert system is invoked for "can" and "should" questions and a natural language generator is used to provide the actual English of the explanation.

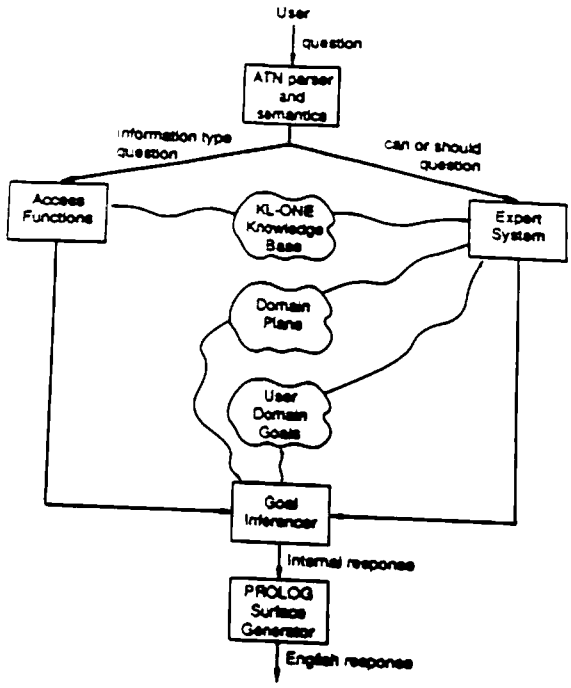

Figure 1: ADVISOR System Organization

3 Influence on Content

Our focus is the influence of the user's goal in the underlying problem domain on explanation content. ADVISOR uses goals[1] to influence explanation content in two ways. Alternative explanations of the same piece of advice (i.e., *yes*) can be generated in response to a given question for different goals. Goals can also influence the advice itself and thus it can also produce different responses to the same question given different goals. In this paper, we use a pair of examples where the advice is different.

Suppose a situation where two users are deciding whether to take the course *Artificial Intelligence*. In both cases, the users are first semester juniors, have taken all the prerequisites of *Artificial Intelligence*, and they will be able to complete the computer science major in time for graduation. Now add to the scenario two different goals each

[1]These goals are derived by ADVISOR from a sequence of questions. See [7] for a description of how this is done.

user has for selecting courses. User 1 wants to take courses in the normal sequence as recommended by the department. User 2 wants to take courses that maximize his/her personal interest in the field of artificial intelligence. These are represented in the system as *(plan c-user c-normal-sequencing)* and *(plan c-user concentrate-on c-ai)* respectively.

ADVISOR generates different advice and justifications for the two users as shown in Figure 2. Since there is flexibility on when the course can be taken, taking it early would allow the student to take more artificial intelligence courses if desired, but it can equally well be taken later if other constraints dictate.

User 1: *plan c-user normal-sequencing*

You should not take Artificial Intelligence. I assume that you want to take courses in the normal sequence. Artificial Intelligence is a second term junior course. You have not taken Digital_logic and Computability. They are first term junior courses. You have not taken Fundamental_algorithms, Discrete_math_2 and Software_lab. They are second term sophomore courses.

User 2: *plan c-user concentrate-on c-ai*

You should take Artificial Intelligence. I assume that you want to concentrate on AI. Artificial Intelligence is an AI course and is required by all other AI courses.

Figure 2: Different Explanations for Different Plans

---

4 Producing an Explanation

To produce a tailored explanation, ADVISOR uses three stages of processing. It first places information that is potentially relevant to the user's goal in working memory. Inferencing is then initiated, producing advice (i.e., either *yes* or *no*) and a hierarchical trace reflecting subgoaling. The trace is passed to the natural language generator, which prunes and reorganizes nodes in the trace as they are realized in English.

Figure 3 shows a compressed version of the trace produced for User 1's explanation before pruning has occurred. The inference trace takes the form of an and-or tree in which or-nodes correspond to backward chaining goals[2]and and-nodes correspond to applications of individual rules to those goals. We note that to determine potentially relevant information, ADVISOR uses a partitioning of its knowledge base into different perspectives. Information about a questioned object is retrieved from a perspective related to the user's goal and placed into working memory. In this case, it would retrieve information about the course, *Introduction to Artificial Intelligence* from the normal course sequence hierarchy. Inferencing is initiated using backchaining from the *should-take* goal with the applicable rule instantiated as in lines 1-5 of Figure 3. It states that a user should take a course if the user *can* take the course[3], if the user's goal is advanced by taking the course (i.e., that goal would be better satisfied by taking it), if it is not the case that the user should not take the course (e.g., there is room in the user's schedule or this course is better than some other courses in the user's schedule) and if the user's schedule will permit timely graduation. Thus, note that the inferencer uses rules that state how a particular action will help achieve a user plan. For Explanation 1, the user's plan to take courses in the normal sequence is advanced if the user has completed all courses up to the suggested second semester junior level (C-J-2) since artificial intelligence is a second semester junior level course. (For the goal on line 10 of the trace, the applicable rule gives rise to the preconditions shown on lines 15-17, Figure 3). More details on how ADVISOR produces the inference trace of a tailored explanation are provided in [8].

---

[2]In this paper, we speak of *user* goals as well as *inferencer* goals. The inferencer reasons with *plans* to satisfy user goals.

[3]Regardless of whether the user's queried action helps to achieve his/her goal, if it is not permissible or will prevent the student from completing the major, the advice is always negative. Rules encoding such absolute constraints include "a course cannot be taken before its prerequisite", or "a course should not be taken if it prevents the student from completing requirements by the time s/he is a senior". Joshi et al [5] specify how to generate appropriate responses for different cases when the response is negative. In our work, we have focused more on providing different goal-oriented advice for positive advice.

```
GOAL: (SHOULD-TAKE C-USER C-INTRO-TO-AI) => NIL                                [01]
COND: (CAN-TAKE C-USER C-INTRO-TO-AI)                                          [02]
      (ADVANCES-PLAN C-USER C-INTRO-TO-AI)                                     [03]
      (~ (SHOULD-NOT-TAKE C-USER C-INTRO-TO-AI))                               [04]
      (SCHEDULE-OK C-USER) => NIL                                              [05]

  1. GOAL: (CAN-TAKE C-USER C-INTRO-TO-AI) => T                                [06]
     COND: (NOW-OFFERED C-INTRO-TO-AI)                                         [07]
           (SATISFIED-PREREQS C-USER C-INTRO-TO-AI)                            [08]
           (~ (TAKEN C-USER C-INTRO-TO-AI)) => T                               [09]

     ***** we omit inferencing for subgoals of the can-take goal  *****

  2. GOAL: (ADVANCES-PLAN C-USER C-INTRO-TO-AI) => NIL                         [10]
     COND: (PLAN C-USER FULFILL-REQUIREMENTS)                                  [11]
           (SUPERC C-INTRO-TO-AI C-REQUIRED)                                   [12]
           (ANNOTATION (PRECURSOR-OF-MAJOR C-INTRO-TO-AI)) => NIL              [13]

       1. GOAL: (PLAN C-USER FULFILL-REQUIREMENTS) => NIL                      [14]

     COND: (PLAN C-USER NORMAL-SEQUENCING)(INDIVIDUATES C-J-2 C-SEMESTER)      [15]
           (SUPERC C-INTRO-TO-AI C-J-2)                                        [16]
           (COMPLETED-UP-TO C-USER C-J-2) => NIL                               [17]

       1. GOAL: (PLAN C-USER NORMAL-SEQUENCING) => T                           [18]
       2. GOAL: (INDIVIDUATES C-J-2 C-SEMESTER) => T                           [19]
       3. GOAL: (SUPERC C-INTRO-TO-AI C-J-2) => T                              [20]
       4. GOAL: (COMPLETED-UP-TO C-USER C-J-2) => NIL                          [21]
          COND: (IMMEDIATELY-PRECEDES C-J-1 C-J-2)                             [22]
                (COMPLETED C-USER C-J-1)(COMPLETED-UP-TO C-USER C-J-1) => NIL  [23]

            1. GOAL: (IMMEDIATELY-PRECEDES C-J-1 C-J-2) => T                   [24]
            2. GOAL: (COMPLETED C-USER C-J-1) => NIL                           [25]
               COND: (TAKEN C-USER C-DIGITAL-LOGIC)                            [26]
                     (TAKEN C-USER C-COMPUTABILITY) => NIL                     [27]

                 1. GOAL: (TAKEN C-USER C-DIGITAL-LOGIC) => NIL                [28]
                 2. GOAL: (TAKEN C-USER C-COMPUTABILITY) => NIL                [29]

            3. GOAL: (COMPLETED-UP-TO C-USER C-J-1) => NIL                     [30]
               COND: (IMMEDIATELY-PRECEDES C-SO-2 C-J-1)                       [31]
                     (COMPLETED C-USER C-SO-2)                                 [32]
                     (COMPLETED-UP-TO C-USER C-SO-2) => NIL                    [33]

                 1. GOAL: (IMMEDIATELY-PRECEDES C-SO-2 C-J-1) => T             [34]
                 2. GOAL: (COMPLETED C-USER C-SO-2) => NIL                     [35]
                    COND: (TAKEN C-USER C-FUNDAMENTAL-ALGORITHMS)              [36]
                          (TAKEN C-USER C-DISCRETE-MATH-2)                     [37]
                          (TAKEN C-USER C-SOFTWARE-LAB) => NIL                 [38]

                      1. GOAL: (TAKEN C-USER C-FUNDAMENTAL-ALGORITHMS) => NIL  [39]
                      2. GOAL: (TAKEN C-USER C-DISCRETE-MATH-2) => NIL         [40]
                      3. GOAL: (TAKEN C-USER C-SOFTWARE-LAB) => NIL            [41]

                 ***** we omit recursive inferencing for earlier semesters  *****
```

Figure 3: Inference trace

4.1 The Natural Language Generator

Our approach to explanation for expert systems emphasizes use of natural language generation techniques in place of traditional templates. ADVISOR'S explanation generator, written in Prolog, consists of two main modules, a *concept dictionary* and a Declarative Clause Grammar (DCG) language generator [4]. The explanation generator calls the concept dictionary to map the tree-structured inference trace produced by the expert system into a linear sequence of propositions which the DCG generator can then render into English. The explanation generator traverses the inference trace in depth-first fashion, pruning portions and linearizing the remainder.

The concept dictionary consists of entries for every goal predicate that can occur in an inference trace. Whenever an inference trace node is encountered during the traversal, the concept dictionary entry corresponding to its predicate is invoked. The entries control the processing of nodes in the inference trace and choose case frames to express those nodes. Each case frame represents a single proposition, and consists of a predicate and its arguments. Individual predicates and arguments are realized as verbs and their case roles in a variety of ways according to context.

The concept dictionary entries collectively embody the various principles that dictate when and how nodes of the trace are removed or reordered for language production. An entry decides whether a node will be realized in the final explanation and whether its descendents should be explored. It also determines the order in which its subtree will be explained and can override the normal depth-first traversal.

We distinguish four criteria for excising portions of the trace. Excised nodes may be *goal unrelated, redundant, silent* or *semantically implied.* In addition, a node entry may override the normal depth first presentation of its subtrees in the trace, causing a reordering of propositions in the explanations. Each of these cases is explained in more detail below.

4.1.1 Pruning Goal Unrelated Nodes

Processing of a node by a concept dictionary entry is influenced by the node's truth value, as well as the identities and truth values of its parents and children. ADVISOR uses several general principles encoded as tests in node entries. If a node is goal unrelated (i.e., it does not contribute towards advancing the user's plan), its result is true as well as its parent's result, and it is an *and* sub-goal (i.e., this node and its siblings must be true in order for its parent to be true), then the node is excised from the explanation. This rule means that background preconditions for pursuing an action are omitted from the explanation when user goal related preconditions for pursuing it are also true. This rule was used in producing explanation 2 of Figure 2. Its inference tree contains 4 subnodes to the root node *(should-take c-user c-intro-to-ai)* as shown in Figure 4. Since all subnodes are true, only the user goal related subnode (i.e., the subtree describing how the user's plan is advanced) is included in the final explanation.

```
GOAL: (SHOULD-TAKE C-USER C-INTRO-TO-AI) => T
COND: (CAN-TAKE C-USER C-INTRO-TO-AI)
      (ADVANCES-PLAN C-USER C-INTRO-TO-AI)
      (~ (SHOULD-NOT-TAKE C-USER C-INTRO-TO-AI))
      (SCHEDULE-OK C-USER) => T
```

Figure 4: Should-take rule from Explanation 2's trace

On the other hand, if a node has two or more *or* subnodes, then subnodes that were proved false are excised from the tree. This principle is followed in removing the node shown in lines 11-13 from the trace for explanation 1 (See Figure 3), where the inferencer unsuccessfully attempted to satisfy the *advances-plan* goal in line 12 with a rule for another plan. Finally, if an and subgoal is proved false, remaining siblings and their subgoals are never pursued in the inferencing process. We are currently modifying this so that when the final is answer is "no", we do indicate all reasons for its failure.

4.1.2 Pruning Redundant Nodes

Redundant nodes arise when the same subgoal appears several times in the same inference. While this redundancy may be essential for correct inference results, ADVISOR may nonetheless assume in some circumstances that the information need not be repeated in its explanation. That is, ADVISOR doesn't provide the entire logical basis for a conclusion -- it obliges the user to complete part of its explanation with fact(s) mentioned earlier in the overall explanation. Identification of redundant nodes is, of course, context-sensitive, and is supported by a history which the explanation generator maintains during traversal. For example, when a course fails to support the student's goal, say to concentrate on AI, ADVISOR points out exactly how it fails to do so by relating subtrees of the trace which show that the course is not an AI course, and that it is not a prerequisite to an AI course, among other things. Although a node identifying the student's goal appears in each of these subtrees, for brevity, ADVISOR only mentions the student's goal once.

4.1.3 Pruning Silent Nodes

Silent nodes are nodes that, while essential for logically founded inferencing, are irrelevant for natural language explanation. They are usually leaves of the inference trace that represent obvious propositions which ADVISOR assumes every user knows. For example, the inferencer's production rules sometimes include preconditions whose sole purpose is to ensure correct binding of variables in other preconditions of the production. One such precondition, when instantiated, is *(individuates c-j-2 c-semester)*, shown in line 19 of the trace for explanation 1 (Figure 3). It could be stated in English as ''Second semester of junior year is a semester''. Clearly, however, one would not need to say this to a student. Nodes such as these are handled by a default concept dictionary entry that returns a null list of case frames. This is similar to Swartout's use of viewpoints [9] to suppress implementation details.

4.1.4 Pruning Semantically Implied Nodes

Finally, a verb selected to translate the predicate of a node may semantically imply the content of the node's subtree. A dictionary entry for a verb indicates the possible subtrees it implies. For example, in the explanation for user 2 in Figure 2, the sentence "All other AI courses require it" is triggered by the concept dictionary entry for the "first-in-area" node and the details of the node's subtree are not presented.

4.1.5 Reordering Subnodes

Sometimes part of the inference trace is best presented in an order which varies from the depth-first order of traversal to highlight user goal related information. Reordering may emphasize the relation between a stated fact and the goal or may move goal related information towards the beginning of a text sequence. For example, the following recursive rule is used when the student wants to take courses in the normal sequence to see if s/he has taken the courses from earlier semesters:

```
(RULE-14  COMPLETED-PREVIOUS-SEMESTERS
          (IF (immediately-precedes ?sem1 ?sem2)
              (completed c-user ?sem1)
              (completed-up-to c-user ?sem1))
          (THEN (completed-up-to c-user ?sem2)))
```

Beginning with the semester when the course in question is usually taken, recursive invocations apply to successively earlier semesters. Depth-first traversal of the resulting inference trace will encounter nodes for those semesters in reverse chronological order, but ADVISOR presents them chronologically. On the way down, ADVISOR saves courses that haven't been taken. For each COMPLETED node on the way back up, if the student is missing course(s) from that semester ADVISOR does two things:

1. Retrieve and produce propositions of the form "You have not taken X" for each course.
2. Produce extra propositions of the form "X is an Nth-semester-and-year course" to indicate the relevance of statements produced by (1) to the user's goal.[4]

---

[4]These extra propositions include the semester from a COMPLETED node and the course from its subtree.

Thus ADVISOR alternates between courses not taken and the semester they belong to in chronological order. Later, our surface generator combines the propositions for courses not taken during a semester into one proposition, combines the propositions relating the courses to the semester into a second proposition and finally pronominalizes the courses in the second proposition. The end result is the last four sentences of Explanation 1. If straight depth-first presentation on a node-by-node basis were done for this portion of the trace, the explanation would read:

```
... You have not completed the first term junior courses.
    You have not taken Digital_logic and Computability.
    You have not completed the second term sophomore courses.
    You have not taken Fundamental_algorithms, Discrete_math_2
    and Software_lab.
```

Incidentally, if the COMPLETED-UP-TO node immediately under ADVANCES-PLAN were true, ADVISOR would just say "You have taken the preceding courses" and omit the subtree.

5 Related Work

Very little work has been done within the expert system environment on producing explanations that are tailored to the system user. One exception is work by Wallis and Shortliffe [11] who show how to generate different explanations depending on whether the user has expertise in the domain. Their approach varies the amount of detail provided to a user based on the complexity of individual inference rules. Note that domain expertise is a long term user characteristic while user goals may change many times over the course of a conversation. Wallis and Shortliffe are thus addressing a different aspect of user modelling than we are.

Joshi et al [5] show how to generate more cooperative responses to a user as part of advising dialog when his/her underlying goal is not best achieved through the stated plan. Instead of simply responding ''yes'' or ''no'' to the user, they enumerate a number of different cases of how the stated and underlying plans may be related along with associated response types for each case. Van Beek [10] describes an implementation of Joshi et al's algorithm that can produce an internal representation of the response,

although not the actual English. Their work focuses on how to respond when the advice is "no", while ours emphasizes production of justifications for positive responses. Some combination of the two methods would ultimately be desirable in a full system.

6 Conclusions

We have described a model for pruning a tree-structured inference trace of extraneous information so that user goal related information can be highlighted. Our model allows us to remove goal unrelated information under certain circumstances and to remove extraneous information that in other ways can be inferred from the remaining text. Reordering of propositions is also used to highlight information that is related to the user's needs. Highlighting user related advice by removing extraneous information is one important part of producing more convincing explanations.

References

[1] Allen, J. F. and Perrault, C. R.
Analyzing Intention in Utterances.
Artificial Intelligence, 15(1): pages 143 - 178, 1980.

[2] Brachman, R, Bobrow, R., Cohen, P., Klovstad, J., Webber, B. L. and Woods, W. A.
Research in Natural Language Understanding .
Technical Report 4274, Bolt Beranek and Newman Inc., August, 1979.

[3] Clancey, W. J.
The Epistemology of a rule-based expert-system: a framework for explanation.
Artificial Intelligence, 20(3): pages 215 - 251, 1983.

[4] Derr, M.A. and McKeown, K. R.
Using Focus to Generate Complex and Simple Sentences.
In *Proceedings of the 10th International Conference on Computational Linguistics*, pages 501-4. Stanford, Ca., July , 1984.

[5] Joshi, A. , Webber, B and Weischedel, R.
Living Up to Expectations: Computing Expert Responses.
In *Proceedings of AAAI-84.* American Association of Artificial Intelligence, 1984.

[6] Swartout, W. R.
The GIST behavior explainer.
In *Proceedings of the Third National Conference on Artificial Intelligence.* American Association of Artificial Intelligence, Washington, DC, 1983.

[7] McKeown, K. R., Wish, M. and Matthews, K.
Tailoring Explanations for the User.
In *Proceedings of the IJCAI.* International Joint Conferences on Artificial Intelligence, 1985.

[8] McKeown, K. R.
Generating Goal Oriented Explanations.
accepted to International Journal of Expert Systems, Special Issue on Natural Language Processing, 1988.

[9] Swartout, W. R.
XPLAIN: a system for creating and explaining expert consulting systems.
Artificial Intelligence, 21(3): pages 285 - 325, 1983.

[10] van Beek, P.
A Model for generating better explanations.
In *Proceedings of the 25th Annual Meeting of the ACL.* Association of Computational Linguistics, Palo Alto, California, 1987.

[11] Wallis, J.W. and Shortliffe, E.H.
Explanatory Power for Medical Expert Systems: Studies in the Representation of Causal Relationships for Clinical Consultation.
Technical Report STAN-CS-82-923, Stanford University, 1982.
Heuristics programming Project. Department of Medecine and Computer Science.

[12] Woods, W. A.
Procedural Semantics for Question-Answering Machine.
In *Proceedings of the Fall Joint Computer Conference.* AFIPS Press, Montvale, N.J., 1968.

[13] Woods, W. A.
Transition network grammars for natural language analysis.
Communications of the ACM, 13(10): pages 591 - 606, 1970.