Inferring User-Oriented Advice in ADVISOR

Robert A. Weida and Kathleen R. McKeown

Columbia University, Department of Computer Science

INTRODUCTION

To be considered cooperative, an expert system must be easy to interact with. It must produce responses that are contextually appropriate, sensitive to the needs of users and suited to the user's level of sophistication. Expert system responses must be explained in a clear and concise fashion, often to untrained users. We have constructed a student advising system called ADVISOR, which contains a rule-based expert system, to test our ideas in natural language understanding, user modeling, user-oriented explanation, and text generation. ADVISOR assists computer science majors in course selection by providing information and offering advice during a natural language question-answering dialogue.

In previous papers, we presented an overview of ADVISOR and detailed our methodologies for deriving user goals from the discourse and expressing expert system reasoning in natural language that emphasizes user goals (McKeown *et al.*, 1985; McKeown, 1988; McKeown and Weida, 1988). In this paper we focus on how the expert system infers contextually appropriate advice tailored to the user's goals and academic status, complete with supporting justifications and additional relevant observations. Since advice is geared to the individual student, ADVISOR can provide different answers to the same question as well as different explanations for the same answer. Indeed, its advice may change, along with the student's goals, as conversation progresses.

THE ADVISOR SYSTEM

ADVISOR answers many straight-forward factual questions, such as "Which hardware courses are offered in the spring?," by constructing knowledge base queries, executing them, and expressing the results in English. For other questions, such as "Can I take NLP?," ADVISOR must reason with facts in its knowledge base about courses, current course offerings, and the student's academic standing by invoking its underlying expert system. Still other questions, like "Should I take Artificial Intelligence?" are judgmental in nature. For *should* questions in particular, the expert system provides advice geared towards the student's goal, using both rules and knowledge base information to determine how well the course supports the goal. An explanation generator prunes and organizes a trace of the expert system's reasoning so that it can be expressed in English by a surface generator (Derr and McKeown, 1984). ADVISOR's architecture is diagrammed in Figure 1.

An important characteristic of ADVISOR is its pervasive concern with the student's conversational goals. This concern begins with a facility for deriving a variety of plausible goals from each of the student's utterances using Allen and Perrault's rules (Allen and Perrault, 1980) and tracking them during the course of conversation. Student goals are represented in a hierarchical structure called the context tree. ADVISOR deals with several types of student goals:

- fulfillment of required courses
- concentration on a particular subdiscipline of computer science such as artificial intelligence
- adherence to the normal sequence of courses for the computer science major

The expert system is designed to take these goals into account and produce advice that is appropriate to them using goal-oriented rules which reference information from a Kl-one style knowledge base (Brachman *et al.*, 1979). Concepts in the knowledge base are organized into intersecting hierarchies representing different points of view so the rules can assess the relevance of a course to different student goals. The explanation generator, in turn, seeks to emphasize those aspects of the expert system's reasoning which show how the goal will or will not be furthered by the action that the question proposes (McKeown and Weida, 1988).

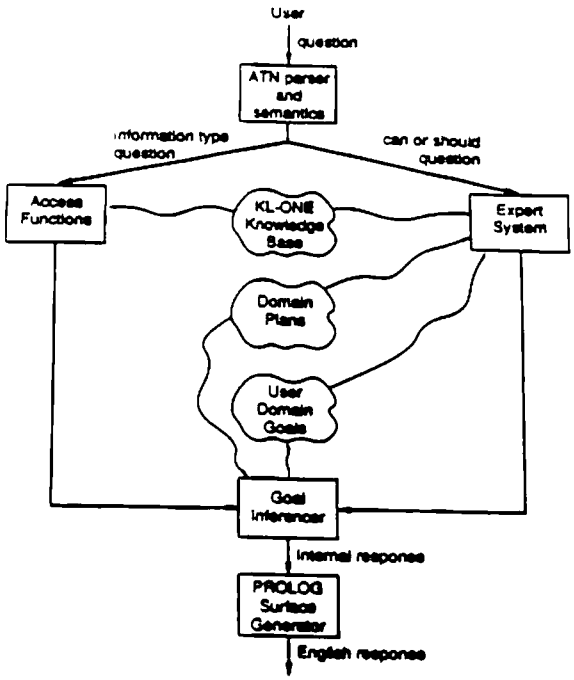

Figure 1: ADVISOR System Organization

ADVISOR has been extensively tested. In the Spring of 1987, students in a graduate course on Natural Language Processing at Columbia University experimented with ADVISOR and evaluated its performance for a class project. During January 1988, computer science students registering at Columbia were given an opportunity to consult with ADVISOR. Almost 40 did so, with encouraging results.

USING A RULE BASE TO INFER USER-ORIENTED ADVICE

We now turn to ADVISOR's rule-based expert system and examine how it infers advice tailored to the user. The expert system determines whether a student *can* or *should* take a certain course in response to student questions. For example:

- Can I take Data Structures?
- Should I take Artificial Intelligence?

These questions are interpreted by ADVISOR's natural language understanding component to produce a speech act and propositional content. Inference goals[1] are extracted from the propositional content and passed along to the expert system for processing. For example:

- (can-take c-user c-data-structures)
- (should-take c-user c-intro-to-ai)

Goals of the latter type are more involved; indeed each *should* question subsumes a *can* question.[2] Accordingly, we will demonstrate our approach via two responses shown in Figure 2 to the question "Should I take Artificial Intelligence?"

[1] In this paper, we speak of *user* goals as well as *inference* goals. The expert system reasons with *plans* to satisfy user goals.

[2] That is, ADVISOR will not recommend a course the student cannot take.

Example 1: *Student plans to take courses in the normal sequence:*

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.

Example 2: *Student plans to concentrate on 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

In the remainder of this section, we will consider the nature of cooperative responses in our domain and show how ADVISOR employs forward and backward chaining to produce them, with particular emphasis on the use of rules to establish a relationship between the course in question and the student's goal. Next, we'll turn to ADVISOR's technique for pointing out extra information related to its advice. Then we will briefly describe the expert system's output and how the explanation generator translates it into English. Finally, we'll outline recent extensions to the expert system.

Cooperative Reasoning

The expert system's response exhibits numerous aspects of cooperative advice embodied in the rule to decide whether a student should take a given course:

```
(RULE-05  SHOULD-TAKE-COURSE
          (IF (can-take c-user ?course)
              (advances-plan c-user ?course)
              (~ (should-not-take c-user ?course))
              (schedule-ok c-user))
          (THEN (should-take c-user ?course)))
```

Each of its four subgoals has implications for user-oriented explanation:

- The first subgoal triggers rules to test a course for conformance with university regulations. A student can take a course if it is being offered, if s/he has taken the prerequisites, and if s/he hasn't already taken it.[3] Later we will outline an approach for suggesting actions that would enable the student to take the course in the future if s/he can't take it now.
- Rules applicable to the second subgoal assess the relevance of a course to the student's plan. These rules are central to our method and several examples are considered below.
- For the third subgoal, rules judge the course within the context of other courses the student wants to take. Even when it furthers the student's plan, the student should not take a course if scheduling constraints force a choice and the alternatives are more expedient.
- The final subgoal treats the long-range scheduling impact of taking a course. A course will not be recommended if taking it would delay graduation, since that shortcoming presumably overrides any benefits.

[3]Here we make simplifying assumptions. Other considerations, e.g., time conflicts, are interesting from the perspective of generating facile explanations. Our expert system does recognize time conflicts, but ADVISOR cannot yet describe them in English.

```
(RULE-06  CONCENTRATE-ON-AREA-DIRECT
          (IF (plan c-user concentrate-on ?area)
              (superc ?course ?area)
              (annotation (first-in-area ?course ?area)))
          (THEN (advances-plan c-user ?course)))

(RULE-07  CONCENTRATE-ON-AREA-INDIRECT

          (IF (plan c-user concentrate-on ?area)
              (superc ?future-course ?area)
              (individuates ?future-course c-course)
              (precursor ?future-course ?course)
              (annotation (first-in-area ?future-course ?area)))
          (THEN (advances-plan c-user ?course)))

(RULE-10  CONCENTRATE-ON-AREA-TOPICS
          (IF (topics ?course ?topic)
              (superc ?topic ?area)
              (plan c-user concentrate-on ?area))
          (THEN (advances-plan c-user ?course)))

(RULE-16  NORMAL-SEQUENCING
          (IF (plan c-user normal-sequencing)
              (individuates ?sem c-semester)
              (superc ?course ?sem)
              (completed-up-to c-user ?sem))
          (THEN (advances-plan c-user ?course)))
```

Figure 3: Rules for Reasoning with Plans

Notice in the examples of Figure 2 that ADVISOR's explanation generator has emphasized the student's plan by eliminating some of these subgoals from the English explanation because they are comparatively unimportant.

The Inference Process

ADVISOR's expert system is organized along traditional lines, with an inferencer that reasons through production rules, based on a working memory of assertions. Assertions about the user's academic standing, such as chronological progress in the major, courses already taken, etc., are extracted from the static portion of ADVISOR's user model residing in the knowledge base. The student in our examples is a second semester junior who has already taken the prerequisites for Artificial Intelligence, as well as enough required courses to permit timely graduation. The context tree, which is the dynamic portion of our user model, contributes assertions about the student's plans. By default, the expert system assumes that s/he plans to take courses in the normal sequence.

For *should* questions, an initial forward chaining process constructs a tentative schedule of all required courses which the student must still take, distributed over the semesters remaining until graduation. Scheduling is geared to the student's particular plan. When the student's plan is to concentrate on AI, this process attempts to leave room for AI electives during the current semester. When the plan is to take courses in the normal order, required courses are scheduled to keep pace with usual practice. The schedule is realized as a set of assertions which augment working memory so that subsequent analysis can consider a course in terms of its scheduling impact.

After the provisional schedule is complete, the inferencer computes a yes-no response to the student's query by backward chaining: using rules to recursively reduce goals into conjunctive subgoals that must be ultimately satisfied by working memory assertions. The *should-take-course* rule is designed to determine the possibility of taking a course and, more interestingly, the advisability of doing so in light of the student's plans.

Plan-specific rules like those in Figure 3 represent ways to achieve student plans. When triggered by working memory assertions about the student's plans, they heuristically relate courses to plans using information drawn from the relevant knowledge base hierarchy. Naturally a course may support other

plans which the student does *not* have, but since there are no working memory assertions to trigger rules for those plans, only pertinent information will be considered.

A fragment of the knowledge base relevant to our examples is diagrammed in Figure 4. The concepts are organized into intersecting hierarchies intended to represent alternate points of view and support alternative explanations. There are is-a hierarchies for topics (AI, hardware, software, theory), requirements/electives, and the normal sequence of courses by semester (freshman-1, freshman-2, ... , senior-2). The hierarchies show how courses relate to specific user plans. The expert system also uses the hierarchies to circumscribe the knowledge base with respect to the plan, thereby limiting the portion of the knowledge base which must be processed. Consider Artificial Intelligence, a required AI course usually taken during the latter semester of the junior year. The *c-intro-to-ai* concept appears simultaneously in the topics hierarchy under the *c-ai* concept, in the requirements/electives hierarchy under *c-required*, and in the normal sequence hierarchy under *c-j-2*. Depending on the student's plan, any one of these relationships may be used to recommend taking Artificial Intelligence.

In Example 1, the student plans to take courses in the normal sequence, so rule 16 applies to the *advances-plan* subgoal introduced by the *should-take-course* rule. Rule 16 references the normal sequence hierarchy[4] to compare the course's normal chronological position with the set of courses actually taken in previous semesters. For the *completed-up-to* subgoal, the appropriate subordinate rules fire, consulting assertions from the normal sequence hierarchy by semester to discover which required courses are usually taken earlier and whether the student has, in fact, taken all of them.

In Example 2, the student plans to concentrate on AI. The first three rules shown in Figure 3 test the relationship between a course and a given subdiscipline of computer science. If, as in this instance, the student wishes to concentrate on AI, the rules indicate that s/he should take AI courses (rule 6), their direct or indirect prerequisites (rule 7), or other courses which cover subjects relevant to AI (rule 10). Since Artificial Intelligence is an AI course, rule 6 is successfully applied to the *advances-plan* subgoal of the *should-take-course* rule.

Note that in more complex cases where the student's schedule is full, the course in question must be compared to the alternatives. ADVISOR has rules to compare courses according to each type of plan.

Volunteering Information with Annotations

A cooperative advisor points out relevant information even when it isn't strictly necessary to answer a question. We have introduced a facility to include such observations directly in the inference trace: the subgoals of specific rules can be augmented with distinguished subgoals we call *annotations* corresponding to observations which, if true, would be appropriate to make at that juncture in the explanation. These annotations are evaluated in the standard way, but they have no effect on the result of the encompassing inference. This isolation is accomplished by means of a distinguished node which subsumes the annotation's subtree and is always true, regardless of the annotation's success or failure.

Annotations are currently used to point out additional facts about certain courses depending on the student's plan. In Example 2, using the annotation in rule 6, ADVISOR mentions that "... [Artificial Intelligence] is required by all other AI courses." Similarly, ADVISOR can observe that Data Structures must precede all other required courses. A potential use of annotations would be to identify specific examples that illustrate general principles, e.g., "You cannot take Artificial Intelligence. You must first take a course that covers LISP, say Intermediate Programming."

Inference Trace

The expert system's output consists of yes/no advice along with a supporting explanation in the form of a rule invocation trace which captures the backward chaining process.[5] The trace is recorded during

---

[4] The *superc* predicate expresses hierarchical relationship.

[5] The forward chaining process is not directly accessible to the explanation generator, but its residual effects are apparent in the trace.

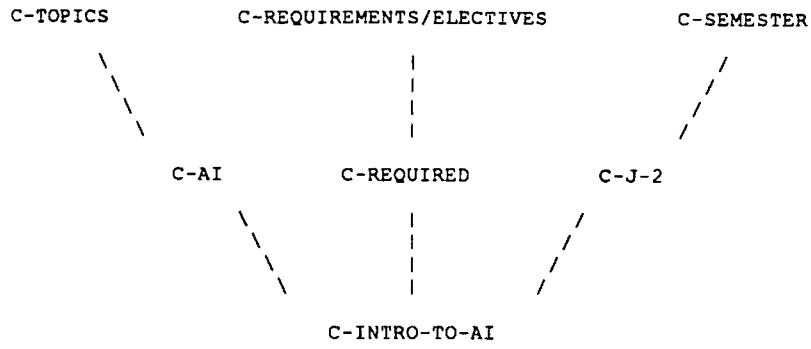

Figure 4: A Fragment of the Knowledge Base

inferencing in a hierarchical data structure which reflects the and/or tree implicit in backward chaining. It consists of alternating levels of or-nodes (corresponding to goals) and and-nodes (corresponding to rule invocations).

In most expert systems, backward chaining is carried out strictly to decide if a goal is satisfiable. Thus, when a subgoal introduced by a rule fails, the system need not consider the remaining subgoals. In fact, for reason of efficiency, subgoals are often carefully arranged so that searching is usually cut off as quickly as possible. However, the explanation task raises another issue: it may be helpful for an advisor to specify several reasons why an inference goal fails, e.g., "Expert Systems isn't offered this semester. Besides, you haven't taken the prerequisite." To support such explanations, the inferencer must sometimes continue processing a set of subgoals after one of them fails. We could have it pursue all subgoals at all times regardless of their success or failure without changing the success or failure of the top level goal, but this would make backward chaining inefficient and would also confront the explanation generator with a vastly larger inference trace. Our pragmatic solution is to tag individual rules for which all subgoals should be pursued, and we are currently modifying the inferencer and rules accordingly. The resulting inference will embody conceptually complete explanations without unnecessary clutter.

The explanation generator (McKeown and Weida, 1988) fashions a response from the inference trace, emphasizing user concerns by removing extraneous information and ordering the remainder. (The response also varies according to user sophistication.) The explanation generator emits a linear sequence of case frame propositions which the surface generator then renders into English.

Ongoing Work

We have extended the expert system in two directions: it can now handle simultaneous plans conjunctively as well as disjunctively, and it can introduce hypothetical situations for consideration at any point during backward chaining. These enhancements are not yet reflected in ADVISOR's English output.

Our plan derivation algorithm recognizes concurrent user plans underlying the discourse. By default, the inferencer treats them disjunctively. Indeed, a course can generally be recommended if it satisfies any one of the student's plans. We have implemented forward chaining rules for scheduling required courses with respect to conjunctive plans as well as backward chaining rules for rating and comparing courses based on conjunctive plan satisfaction, but we must also extend our plan derivation algorithm (McKeown *et al.*, 1985) to address the difficult problem of judging when multiple plans attributed to the student are conjunctive and when they are disjunctive. This distinction raises interesting questions for natural language explanation and we are eager to pursue them in the future.

Another important aspect of advice involves consideration of hypothetical situations. For instance, a student may ask "Can I take Artificial Intelligence next semester if I take Data Structures this semester?". Hypothetical assertions on the part of the student may be specified as input together with the query.

Similarly, hypothetical reasoning could be used within an annotation to volunteer remedies when replying negatively to questions like "Can I take Artificial Intelligence?," e.g., "No, but you can take it next semester if you take Data Structures now."[6]To facilitate this, rules may now specify hypothetical assertions, typically as part of an annotation. The inferencer supports hypothetical reasoning by allowing hypothetical assertions and retractions of facts in working memory at arbitrary levels of the backward chaining process.

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 (Wallis and Shortliffe, 1982) 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 modeling than we are.

Joshi et al (Joshi *et al.*, 1984) 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 (van Beek, 1987) 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.

CONCLUSION

We have described how ADVISOR's expert system uses a student's conversational goals and academic standing to decide whether s/he should take a particular course, thus exhibiting user sensitivity along two dimensions. The inferencer determines advice by reasoning with production rules that integrate both kinds of information about the user with information about the computer science major. An inference trace records the justification for the advice. From the trace, ADVISOR's explanation generator can fashion convincing English explanations tailored to the student's needs.

REFERENCES

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

Brachman, R. Bobrow, R., Cohen, P., Klovstad, J., Webber, B. L. and Woods, W. A. (August 1979). *Research in Natural Language Understanding* (Tech. Rep. 4274). Bolt Beranek and Newman Inc.,

Derr, M.A. and McKeown, K. R. (July 1984). Using Focus to Generate Complex and Simple Sentences. *Proceedings of the 10th International Conference on Computational Linguistics*. Stanford, Ca..

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

McKeown, K. R. (1988). Generating Goal Oriented Explanations. accepted to International Journal of

[6]Joshi et al (Joshi *et al.*, 1984) specify how to generate appropriate responses for different cases when the response is negative.

Expert Systems, Special Issue on Natural Language Processing.

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

McKeown, K. R. and Weida, R. A. (1988). Highlighting User Related Advice. submitted to Seventh National Conference on Artificial Intelligence.

van Beek, P. (1987). A Model for generating better explanations. *Proceedings of the 25th Annual Meeting of the ACL.* Palo Alto, California: Association of Computational Linguistics.

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