Exploiting user feedback to compensate for the unreliability of user models

Exploiting User Feedback to Compensate for the Unreliability of User Models

JOHANNA D. MOORE *
University of Pittsburgh
Department of Computer Science
and
Learning Research and Development Center
Pittsburgh, PA 15260, USA

and

CÉCILE L. PARIS **
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695, USA

(Received 14 October, 1990; in final form 9 April, 1992)

Abstract. Natural Language is a powerful medium for interacting with users, and sophisticated computer systems using natural language are becoming more prevalent. Just as human speakers show an essential, inbuilt responsiveness to their hearers, computer systems must "tailor" their utterances to users. Recognizing this, researchers devised user models and strategies for exploiting them in order to enable systems to produce the "best" answer for a particular user.

* Dr. Johanna D. Moore holds interdisciplinary appointments as an Assistant Professor of Computer Science and as a Research Scientist at the Learning Research and Development Center at the University of Pittsburgh. Her research interests include natural language generation, discourse, expert system explanation, human-computer interaction, user modeling, intelligent tutoring systems, and knowledge representation. She received her MS and PhD in Computer Science from the University of California at Los Angeles, and her BS in Mathematics and Computer Science from the University of California at Los Angeles. She is a member of the Cognitive Science Society, ACL, AAAI, ACM, IEEE, and Phi Beta Kappa. Readers can reach Dr. Moore at the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260; e-mail: jmoore@cs.pitt.edu.

** Dr. Cecile Paris is the project leader of the Explainable Expert System project at USC's Information Sciences Institute. She received her PhD and MS in Computer Science from Columbia University (New York) and her bachelor's degree from the University of California in Berkeley. Her research interests include natural language generation and user modeling, discourse, expert system explanation, human-computer interaction, intelligent tutoring systems, machine learning, and knowledge acquisition. At Columbia University, she developed a natural language generation system capable of producing multi-sentential texts tailored to the users" level of expertise about the domain. At ISI, she has been involved in designing a flexible explanation facility that supports dialogue for an expert system shell. Dr. Paris is a member of the Association for Computational Linguistics (ACL), the American Association for Artificial Intelligence (AAAI), the Cognitive Science Society, ACM, IEEE, and Phi Kappa Phi. Readers can reach Dr. Paris at USC/ISI, 4676 Admiralty Way, Marina Del Rey, California, 90292; e-mail: paris@isi.edu.

User Modeling and User-Adapted Interaction **2**: 287–330, 1992.

Because these efforts were largely devoted to investigating how a user model could be exploited to produce better responses, systems employing them typically assumed that a detailed and correct model of the user was available *a priori*, and that the information needed to generate appropriate responses was included in that model. However, in practice, the completeness and accuracy of a user model cannot be guaranteed. Thus, unless systems can compensate for incorrect or incomplete user models, the impracticality of building user models will prevent much of the work on tailoring from being successfully applied in real systems. In this paper, we argue that one way for a system to compensate for an unreliable user model is to be able to react to feedback from users about the suitability of the texts it produces. We also discuss how such a capability can actually alleviate some of the burden now placed on user modeling. Finally, we present a text generation system that employs whatever information is available in its user model in an attempt to produce satisfactory texts, but is also capable of responding to the user's follow-up questions about the texts it produces.

Key words: question answering, natural language generation, adaptive systems, text planning, explanation, expert systems, user modeling

1. Introduction: The Need for and Limitations of User Models

Natural Language (even when limited) is a powerful medium for interacting with users or for providing documentation about a system. In fact, some argue that natural language is "critical for the effective use of expert and advisory systems" (Finin *et al.*, 1986, p. 921), and sophisticated computer systems using natural language are becoming more prevalent. However, one of the reasons that natural language is so powerful is that human speakers show an essential, inbuilt responsiveness to their hearers. Therefore, if a computational system is to reap the benefits of natural language interaction, it must similarly "tailor" its utterances to its users. Recognizing this, researchers have devised user models and strategies for exploiting them in order to enable systems to produce the "best" answer for a particular user in one shot. Systems employing such models have convincingly demonstrated that a user model can be used to guide the generation of utterances that are appropriately tailored to a user's knowledge state and goals (e.g., Appelt, 1985; van Beek, 1986; Chin, 1987; McCoy, 1988; Paris, 1988; Woltz *et al.*, 1990).

Because these efforts were largely devoted to investigating how a user model could be exploited to produce better responses, they typically assumed that a detailed and correct model of the user was available *a priori*, and that the information needed to generate appropriate responses was included in that model. However, hand-crafting a detailed user model for each user is prohibitively time-consuming and error-prone. Moreover, Sparck Jones (1984, 1989) has questioned the feasibility of automatically acquiring the complex and detailed user models assumed by existing generation systems. In practice, the completeness and accuracy of a user model cannot be guaranteed.

Thus, unless systems can compensate for incorrect or incomplete user models, the impracticality of building user models will prevent much of the work on tailoring from being successfully applied in real systems.

In this paper, we argue that one way for a system to compensate for an unreliable user model is to be able to react to *feedback* from users about the suitability of the texts it produces. By feedback we mean follow-up questions evoked by previously generated responses (e.g., "What is an X?", "Why?"), indications that a clarification of a response is desired (e.g., by asking a question again), or simple indications that a response was not understood (e.g., "Huh?", "I don't understand.").

The ability to recover when the user is not satisfied with an explanation will alleviate some of the burden now placed on user modeling. With this capability, a system no longer *requires* a detailed and correct user model in order to supply users with the information they seek. It is not forced to attempt to provide a response that will be understood and that will be the most appropriate in *one shot*. Instead, the system can rely on the user to provide feedback when necessary. By responding to this feedback, the system can overcome the limitations of its user model.

This paper is structured as follows. First, we motivate the importance of user feedback by describing the results of studies of naturally occurring advisory interactions. We then describe an architecture for text generation in which a system employs information in a user model when it is available, but also allows the user to ask questions about the text generated. We identify the requirements that such an approach places on the generation process and present our text planner which satisfies these requirements. We also describe the user model employed by our system and explain how it affects the generation process. We then work through a detailed example to illustrate how this approach can alleviate some of the problems faced by systems that rely on *a priori* user models alone. Finally, we discuss related work and present some directions for future research.

2. The Importance of Feedback from the User

While dialogue participants have some models of their interlocutors, they rarely start with detailed and correct models. Yet, they are capable of communicating effectively, even when their initial model turns out to be incorrect. In conversation, humans often rely on feedback from their hearer (Ringle and Bruce, 1988). They expect their interlocutors to ask further questions, request clarification, or simply provide an indication that something was not understood. With this feedback, speakers are able to provide further information,

[The student and teacher are discussing an assignment to implement an infix calculator using two stacks to keep track of the operators and numbers read in. The student is unsure what data type the entries in the operator stack should be and about whether to store the operator or its precedence on the operator stack.]

STUDENT	OK, I wasn't sure how to make the operator stack.
TEACHER	OK, the operator stack...
STUDENT	Is... it... it be a long integer or something like that?
TEACHER	It can be long integers or characters. Sounds like it would naturally fall into characters. OK? Or we could define capital P-L-U-S to be 1, the word capital M-I-N-U-S to be 2, the word divide, D-I-V-I-D-E, to be 3, and the word mult, capital M-U-L-T, to be 4, and the word E-X-P, for exponentiation to be 5. OK? Then we'll have a variable called operator. OK? And we'll say this. If we read in the character plus sign, then operator equals 1. Right?
STUDENT	Isn't it, isn't it, the plus and the minus 1?
TEACHER	No, that's the precedence, OK? We're gonna assign another variable called precedence. OK? So if we read in the plus sign, we're going to say operator equals capital P-L-U-S, precedence equals 1. We read in the minus sign, we're gonna say operator equals M-I-N-U-S, precedence equals 1. Right?
STUDENT	Oh... uh... uh...
TEACHER	That's one way of doing it. Or, we could do... We have to keep track of what the operator was, OK? And we can also write a routine called "precedence" which will return the precedence of an operator, which says, if operator equals P-L-U-S, return 1. If operator equals minus return 1. OK? So we can call this routine whenever we have an operator.
STUDENT	But you can't put characters on the, on the long integers stack.
TEACHER	It's not advisable. If you want to put characters on the stack, we would declare the stack to be of type character. Which you might want to do...
STUDENT	To have a character...
TEACHER	Sure. And the character stack, that particular stack is the operator stack, right? You read in the plus, you put the plus sign on top of the character stack... on the operator stack of type character. So you can do it anyway you feel comfortable with it. But hopefully it should work.
STUDENT	In the array, for each element, do you store the operator or the precedence?
TEACHER	You store the operator.
STUDENT	Only?
TEACHER	Well, you can find out the precedence, right? From any operator you can find out the precedence. OK? From the precedence, can you find out the operator?
STUDENT	No.
TEACHER	Right.

Fig. 1. Sample human–human dialogue.

often based on what they have already said. This phenomenon is particularly illustrated in advisory interactions, such as the one shown in Figure 1, collected by Moore.

In this dialogue, a student and teacher are discussing an assignment to implement an infix calculator using two stacks to keep track of the operators and numbers read in. In this portion of the dialogue, the student asks how to create the operator stack. The student appears to be confused about what data type, long integers or characters, would be most appropriate. The instructor first suggests using a character array and then describes a method of associating numbers with operators so that an integer array could be used. This causes the student to ask a follow-up question indicating that he has confused the numbers used to encode the operators and the numbers indicating the precedence of operators. Later the student asks a vaguely articulated question "Oh... Uh... uh..." and the teacher responds by offering another method of solving the problem. The student then asks another follow-up question to resolve an incompatibility between his belief (that the precedence must be stored) and the instructor's response (that implies it need not be stored.) The instructor elaborates by giving a reason to justify his previous response. Clearly, the instructor does not have a complete model of his listener; if he had, he would have anticipated the listener's need for the elaborated explanation and given it the first time around.

Yet, as the dialogue illustrates, the advice-seeker and the expert are able to communicate. This is because, when hearers do not fully understand a response, they ask follow-up questions, requesting clarification, elaboration, or re-explanation of the expert's response. These follow-up questions are not necessarily well-formulated, as people cannot always pinpoint just what it is they do not understand. In such cases, the follow-up question is vaguely articulated in the form of mumbling, hesitation, repeating the last few words of the expert's response, or simply stating "I don't understand." In the sample dialogue it takes the form of "Oh... uh... uh..." Often the expert does not have much to go on, but must still provide further information, again relying on the hearer to ask more questions if necessary.

From our analysis of dialogues such as this one (Moore, 1989a), we concluded that experts do not have detailed and correct models of advice-seekers. Other researchers, e.g., (Falzon, 1987; Cahour, 1990, 1991), have shown how experts build a user model from a dialogue. At the beginning of an interaction, experts quickly assign a stereotype to their interlocutor, and their answers are driven by the chosen stereotype. As the dialogue proceeds, they refine this stereotypic model, and adapt their answers appropriately. Furthermore, if their interlocutor's reactions to their answers indicate that

they made a mistake in assigning the stereotype, experts change the user model and correspondingly change their behavior.

These studies suggest that, in order to communicate effectively, a system must be able to recognize when the user's reactions indicate that the system's responses are unsatisfactory, and be able to respond appropriately. These capabilities increase the robustness of a system in three ways: a system can start to communicate by generating responses with incomplete or incorrect information in the user model, and await feedback from the user. The user model can then be built or refined from interactions with the user, employing techniques for user model acquisition suggested in several recent works, e.g., (Kobsa, 1984; Sleeman, 1985; Chin, 1989; Lehman and Carbonell, 1989; Mastaglio, 1990; Bunt, 1990; Quilici, 1990; Kass, 1991; Shifroni and Shanon, 1992; Wu, 1991). Finally, the system can recover from an inaccurate user model by addressing the user's follow-up questions.

3. A System That Employs Feedback

We have constructed an advice-giving system that participates in explanatory dialogues with its users. This system generates coherent multi-sentential texts, deciding both what to say and how to organize the selected content. It makes use of information in a user model when it exists, but it is not critically dependent on the quality of the information in that model. In particular, it can provide alternative explanations, elaborate on previous explanations, and respond to follow-up questions in the context of the on-going dialogue, even when the user is not very explicit about what aspect of the explanation was not clear.

3.1. EXPLAINER OVERVIEW

Our generation system was developed as part of the Explainable Expert Systems (EES) framework (Neches *et al.*, 1985; Swartout and Smoliar, 1987), a domain-independent shell for creating expert system applications. A key feature of the EES framework is that it provides an explicit representation of the knowledge needed to support explanation of the terminology and reasoning processes used by the expert system. When an expert system is built using EES, a *development history* is created that records the goal structure and design decisions behind the expert system (Chandrasekaran and Swartout, 1991; Swartout *et al.*, 1991). This structure, the domain knowledge base, and the execution trace of the expert system's behavior are all available for use by the explanation facility.

An overview of the generation system (and its relation to the expert system)

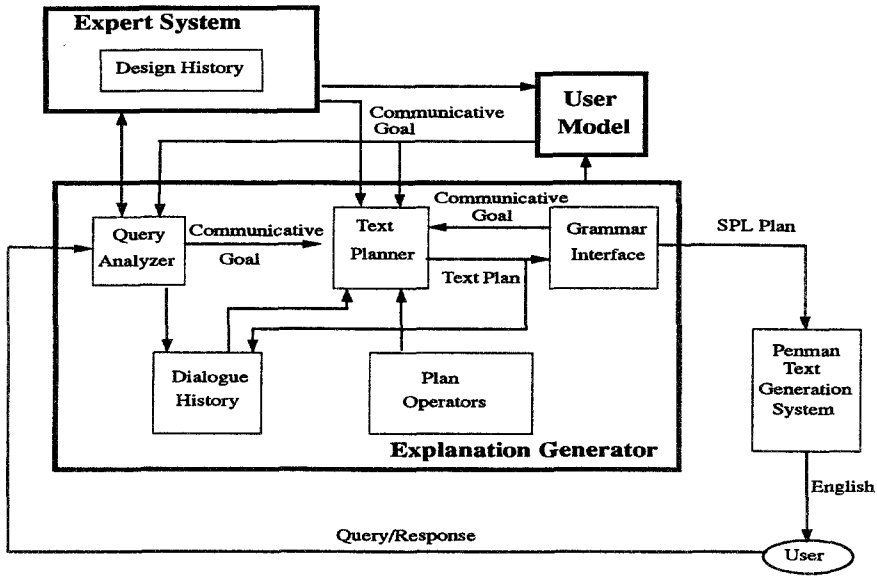


Fig. 2. Architecture of the explanation system.

is shown in Figure 2. To interact with the user, the expert system posts a communicative goal (e.g., get the hearer to adopt the goal of performing an action, make the hearer know a concept, justify a conclusion) to the text planner. The text planner then constructs a text to achieve this communicative goal. As will be discussed in Section 3.2, the system *must understand its own explanations* in order to be able to answer follow-up questions in context and offer elaborating or clarifying explanations. To this end, our system explicitly plans the explanations it produces using a set of explanation strategies. The planning process is recorded to capture the "design" of the explanation.

The explanation is presented to the user and recorded in the *dialogue history*. At this point, the user can pose a question, which will be interpreted by the *query analyzer*. Queries are often follow-up questions regarding the text generated by the system. Even though we assume the user poses queries in a restricted sublanguage, ambiguities may still arise. For example, simple queries such as "Why?" mean different things in different contexts. In our system, context includes the information recorded in the dialogue history, the expert system's state, and the user model. To interpret the user's input, the query analyzer examines these knowledge sources to produce a communicative goal that is then posted to the text planner.

3.2. REQUIREMENTS FOR HANDLING FEEDBACK

Users' questions must be interpreted and answered in the context of the ongoing interaction, and the system's previous explanations make up part of this context. When the user does not fully understand a response, the generation facility must be able to determine what portion of the text failed to achieve its purpose, so that it can clarify misunderstood explanations and elaborate on prior explanations. To provide these capabilities, a system must "understand" the text it generates in terms of what it was trying to convey as well as how that information was conveyed. That is, it must represent and be able to reason about the intentional structure behind an explanation, including the goal of the explanation as a whole, the subgoal(s) of individual parts of the explanation, and the rhetorical means used to achieve these goals (Moore and Paris, 1989; forthcoming).

Our system achieves this understanding by explicitly representing the "design" of the explanations it produces. The system constructs explanations to achieve its communicative goals, recording all of its decisions in a *text plan.* This text plan includes the intentional and rhetorical structure of the text being produced, as well as any assumptions about the user's goals and knowledge that were made while planning the response. This is an important aspect of the system: since we cannot rely on having complete user models, the system may have to make assumptions about the hearer in order to use a particular explanation strategy. Such assumptions are recorded in the text plan and available for later reasoning if there is any indication that a communication failure has occurred.

Text plans thus provide the context necessary to interpret follow-up questions and recover when feedback from the user indicates that the system's explanation is not satisfactory (Moore and Swarout, 1989). When the user indicates that an explanation was unsatisfactory, the system reasons about the text plan that produced the explanation in order to determine the focus of attention, and to decide which of the system's communicative goals might have failed, or which of the assumptions made may have been erroneous. The text plans recorded in the dialogue history are also used to guide the planning of elaborating and clarifying responses (Moore, 1989b) and to avoid repeating information that has already been communicated (Moore and Paris, 1989).[1]

We now present the system in more detail and describe the user model employed in the system.

---

[1] Furthermore, we have also demonstrated how these text plans can be used to select a perspective when describing or comparing objects (Moore, 1989a).

3.3. THE TEXT PLANNER

The text planner constructs a text to achieve the system's communicative goals. In our plan language, communicative goals are represented in terms of the effects that the speaker intends the text to have on the hearer's beliefs or goals. For example, the speaker may intend that the hearer believe some proposition, know about some concept, have some goal, or perform a certain action.[2] To satisfy communicative goals, the planner makes use of explanation strategies that map communicative goals to the linguistic resources for achieving them.

To enable the system to produce coherent multi-sentential texts, these linguistic resources include knowledge about the rhetorical relations defined in Rhetorical Structure Theory (RST) (Mann and Thompson, 1987), a theory of text coherence. RST defines a set of approximately 25 relations (e.g., MOTIVATION, EVIDENCE) that may exist between adjacent spans in a coherent English text. The definition of each relation specifies constraints on the two spans of text being related, as well as the effect on the hearer's beliefs, desires, or intentions that this relation may be used to achieve.

Explanation strategies are encoded in plan operators. Each operator consists of:

- *an effect:* a characterization of the goal that this operator can be used to achieve. An effect may be a communicative effect, such as "hearer believes a proposition", or a linguistic effect such as "provide motivation for an act" or "inform user of a proposition".
- *a constraint list:* a list of conditions that "must" be true before the operator can be applied. Constraints may refer to facts in the system's domain knowledge bases, information in the user model, information in the dialogue history, or information about the evolving text plan.
- *a nucleus:* a subgoal to express the main topic. Every operator must contain a nucleus.
- *satellites:* additional subgoal(s) that will lead to the inclusion of information needed to achieve the effect of the operator. When present, satellites may be marked as required or optional.

An example plan operator is shown in Figure 3. This operator encodes the knowledge that the communicative goal of persuading the hearer to do an act can be achieved by motivating the act in terms of a user goal. The constraints of this operator indicate that, in order to persuade the hearer to do an act, the

[2] Details of the representational primitives used in our system are beyond the scope of this paper. Interested readers are referred to Moore (1989a) and Moore and Paris (forthcoming). For clarity, we provide English paraphrases of the terminology in our examples.

In Formal Notation:

EFFECT: **(PERSUADED ?hearer (GOAL ?hearer (DO ?hearer ?act)))**
CONSTRAINTS: **(AND (STEP ?act ?goal)**
(GOAL ?hearer ?goal))
NUCLEUS: **(FORALL ?goal**
(MOTIVATION ?act ?goal))
SATELLITES: **nil**

English Translation:

To achieve the state in which the hearer is persuaded to do an *act*,
IF the *act* is a step in achieving some *goal(s)* of the hearer,
THEN motivate the *act* in terms of those *goal(s)*.

Fig. 3. Plan operator for persuading user to do an act.

system should look for domain goals that are shared by the hearer and that the act is a step in achieving.[3] If any such goals are found, the planner will post one or more `MOTIVATION` subgoals.

Using a hierarchical planning mechanism (Sacerdoti, 1975), the text planner constructs text to achieve communicative goals. When a goal is posted, the text planner searches its library of plan operators to find those capable of achieving this goal. To determine whether a plan operator may be applied, the planner must check the operator's constraints. When constraints contain unbound variables, satisfying them will cause the text planner to search the expert system's knowledge bases, the dialogue history, and the user model for acceptable bindings for these variables. This will be discussed further in Section 4.

All the plan operators that can achieve the current goal and whose constraints are satisfied are considered candidate operators. *Plan selection heuristics* enable the planner to choose among them, taking into account the user model, the dialogue that has occurred so far, as well as other factors which are discussed in the following section. Once a strategy is selected, it may in turn post subgoals for the planner to refine. As the system plans explanations to achieve its communicative goals, it keeps track of any assumptions it makes

[3] This plan operator actually contains other constraints that refer to the information contained in the dialogue history and the evolving text plan. These are not relevant to our discussion here and are thus omitted for simplicity. See Moore and Paris (forthcoming).

about what the user knows as well as alternative strategies that could have been used to achieve its goals.

The primitive operators in our plan language are speech acts, such as INFORM, RECOMMEND.[4] Whenever a speech act is posted as a subgoal, the *grammar interface* constructs a specification of this speech act in Sentence Plan Language (SPL) (Kasper, 1989). When text planning is complete, these SPL specifications are passed to the surface generator, Penman (Matthiessen, 1984; Mann and Matthiessen, 1985), which produces English utterances. Transforming a speech act into an SPL specification is currently done in a straightforward manner. The speech act type dictates the mood (e.g., declarative, imperative) of the sentence, the predicate of the proposition to be expressed determines the verb type, and nominal groups are constructed for each concept involved in the speech act. This approach has been taken in many generation systems that plan multi-sentential texts, e.g., (McKeown, 1985; McCoy, 1985; Paris, 1987).

In the process of building an SPL specification, new text planning goals may be posted as side effects. This occurs because the text planner reasons about concepts and processes at an abstract level, in the system's knowledge representation language. Because a single system concept may actually represent a very complex structure (e.g., "transformations that enhance maintainability"), the system must "unpack" this structure to construct a nominal group to express such concepts. It is only when this unpacking is done that the system recognizes that certain concepts (e.g., "maintainability") will be mentioned in the final text.

To provide an informative and understandable text, the system must phrase its utterance using terms the user knows and understands. After having "unpacked" a complex data structure, the system checks the user model to see if each additional term to be expressed is known to the user. If, according to the user model, a term is not known to the user, the system can do one of two things: (1) it can assume that the term is known, generate the text using a simple lexical item and record the assumption that was made in the text plan, or (2) the system can post a goal to define the unknown term to the text planner.[5] This phenomenon is discussed further in the Section 4. In this way, our system can *opportunistically* define a new term when the need

[4] Here we are using the term "speech act" where Appelt (1985) would use "surface speech act".

[5] Bateman and Paris (1989, 1991) are studying the problem of choosing the appropriate syntactic structures and lexical items for a specific user or for classes of users. Their system thus does further planning to express a concept in English. If the phrasing component is unable to phrase a concept in terms the user understands, however, it would return control to the text planner by posting a goal to define a concept.

arises. This is often done in human speech, as illustrated in the following explanation given by a doctor when asked about the possible ways to treat migraine (italics indicate our present concern, not spoken emphasis).[6]

> So, for example, say that you told me that you had three to four headaches, and you weren't sure when they would come in the month, [...] what I would recommend with that frequency is that you should be on something prophylactically. *Prophylactically basically means preventing the headache from occurring before it actually starts.* If you had infrequent headaches, maybe several times a year, where you were quite sure when you were going to have the headaches, then I would recommend more something abortive. *That means that when the headache came on, I would treat you at that point.* I would rather, to help prevent side effects from you having to take a medicine on a daily basis, just try to abort them, if they were infrequent.

Because a new term can be introduced in virtually any statement, it would be unwieldy to include a subgoal for defining a new term in each step of each plan operator that might result in introducing a new term. We believe that a more elegant approach is to post the goal to define a term when the need arises, and work the definition into the evolving text plan according to the rules of discourse as represented in plan operators. This is what happens in our system.

When all subgoals have been refined into SPL specifications, planning is complete. The result of the planning process is a *text plan* for achieving the original communicative goal. It provides an explicit representation of the explanation produced by the system, indicating how parts of the plan are related, and what purposes different parts of the generated explanation serve. This text plan can thus be thought of as a "design record" telling the system what it was trying to explain, how it explained it, what assumptions were made, and what alternative strategies could have been selected at various points during the planning process.

The completed text plan is recorded in the *dialogue history* and passed to the grammar interface which examines the plan tree to determine where sentence boundaries should be placed and which, if any, connective markers should be included in the text. The output of the grammar interface is an ordered list of SPL specifications. These are then passed, a sentence at a time, to the Penman system for translation into English.

---

[6] This example is taken from transcripts gathered by Claudia Tapia and Johanna Moore at the University of Pittsburgh.

After a response has been generated, the system awaits feedback from the user. This feedback may be a follow-up question (e.g., "Why?", "What's the difference between CAR and FIRST?", "What is a generalized variable?"), an indication that the user does not understand the system's response ("Huh?"), an indication that the user has no further questions ("OK"), or a response to a question posed by the system.[7] The text plans in the dialogue history in conjunction with the user model and the expert system's problem-solving state provide the information necessary to allow the query analyzer to choose appropriate interpretations for the user's questions. This context is necessary, as even simple questions (such as "Why?") can often be interpreted differently depending on what the user knows, the information currently available in the problem-solving space, or the content of the earlier discussions (Buchanan and Shortliffe, 1984). A detailed example of this phenomenon can be found in Moore and Swartout (1989).

Having interpreted the user's feedback, the query analyzer either returns control to the expert system, or formulates the appropriate communicative goal and passes it on to the text planner to produce a response. If the text planner must produce a response, it plans text as before, except that it now also uses the text plan for the previous response (as recorded in the dialogue history) to guide its decision process.

3.4. THE USER MODEL

Although we argue that requiring a complete and correct user model is unrealistic, we believe that knowledge about the user is necessary for providing explanations the user will find relevant and understandable. For instance, when attempting to persuade a user to perform an action, the system should motivate the action in terms of the user's goals, as represented by the second constraint in the operator shown in Figure 3 page 296: `(GOAL USER ?goal)`.[8] Knowledge about the user is also important when selecting the most appropriate strategy to produce an explanation. For example, in our system, there are many different strategies for describing a concept to the user. Using one strategy, the system describes an object by stating what superclass it belongs to and describing its attributes and its parts. Another strategy calls for giving examples of the concept being described. Yet another strategy draws an analogy

[7] To aid the user in supplying feedback, we also developed a hypertext-like pointing interface (Moore and Swartout, 1990). Using this interface, the user is allowed to highlight parts of the explanation given by the system to request clarifications or elaborations. When the user selects a portion of the text, a menu containing all the questions that might be asked about this portion at this point in the dialogue is displayed, and the user chooses the appropriate one.

[8] The variable ?hearer is globally bound to USER.

with a similar concept. One of the factors that influences the choice of strategy in a particular situation is what the user knows, i.e., drawing an analogy will only be effective if the user is familiar with a concept similar to the one being described. Similarly, the system should only choose the strategy of giving examples if the user will find the examples illustrative, i.e., the user knows the example concepts. Finally, knowledge about the user is useful in deciding which lexical items and syntactic structures should be used to express a concept in English (e.g., Jameson and Wahlster, 1982; Reithinger, 1987; Bateman and Paris, 1989; Haimowitz, 1990). Therefore, having information about the current user is important in providing meaningful explanations.

However, we do not wish our system to be critically dependent on either the completeness or correctness of the user model. Rather, we take the approach that the system should make use of any information it has available about the user, but be ready to react to feedback from the user indicating that the system's utterance was not satisfactory. The system described here is capable of performing adequately even if it has no information about the user, or if that information does not accurately reflect the user's knowledge and goals. When necessary, the system makes assumptions and relies on the user to ask follow-up questions if the response generated is not sufficient. Our system constructs an initial user model, updates it in simple ways during the interaction, and uses it to guide the text planner. The tasks of building and maintaining the user model are currently done in a straightforward way, as this was not the focus of our work. These tasks are clearly complex, but we believe that the ability to reason about the user's feedback in the context of the text plans recorded in our dialogue history will aid our system in detecting and correcting inaccuracies in the user model. We also believe that integrating the results of recent work on user model acquisition such as Kass (1991) will prove promising.

3.4.1. Representation and content of the user model

In this work, we use an *overlay technique* (Carbonell, 1970; Carr and Goldstein, 1977) to model the user. The user's knowledge and goals are thus assumed to be some subset of the system's knowledge and goals. Using this technique, a user model is *incomplete* if the user knows something that is not indicated in the user model. A user model is *incorrect* if it indicates that the user knows some concept or holds some belief when, in fact, the user does not. Note that we are ignoring another sense in which a user model might be incorrect. In particular, we are ignoring the case in which the user holds some belief that is incompatible with the system's model. Detecting misconceptions has been studied extensively in the area of Intelligent Tutoring systems (e.g.,

Brown and Burton, 1978; Stevens *et al.*, 1979; Sleeman and Brown, 1981; Sleeman, 1983; Wenger, 1987, Chapters 16 and 17) and in Computational Linguistics (e.g., Pollack, 1986a; Quilici *et al.*, 1988). There has also been research in correcting misconceptions once detected (e.g., Mays, 1980; Joshi *et al.*, 1984; McCoy, 1988; Quilici *et al.*, 1988). In our current work, we do not attempt to detect inconsistencies between the user model and the system's knowledge, nor will we try to correct such errors.

In our system, the user model records four types of information about the user: the user's goals, the user's knowledge about methods for achieving goals and performing acts (e.g., the user is competent to perform an act), the concepts the user is familiar with, and facts the user believes (i.e., relations between concepts). The user model thus contains the following types of assertions:

- `(GOAL USER` *g*`)`: The system believes that the user has a goal, *g*.
- `(BEL USER (CONCEPT` *c*`))`: The system believes that the user is familiar with a concept, *c*.[9] This assertion means that the user knows a description of the concept *c*.[10] This assertion does not imply that the user knows any particular attributes of *c*. As will be seen later, our user modeling acquisition component employs inference rules such as those defined in Kass (1991) to gather more information about the user's knowledge.
- `(BEL USER` *p*`)`: The system believes that the user believes some fact, *p*, about the world. *p* can be a fact in the domain model. For example, `(BEL USER (ISA CAR-FUNCTION ACCESS-FUNCTION))` indicates that the user believes that the `CAR` function is an access function. Alternatively, *p* can refer to a fact in the problem-solving knowledge. For example, `(BEL USER (STEP APPLY-TRANSFORMATIONS-THAT-ENHANCE-MAINTAINABILITY ENHANCE-MAINTAINABILITY))` indicates that the user believes that applying transformations that enhance maintainability is a step towards achieving the goal of enhancing maintainability.

---

[9] Throughout the paper, we often paraphrase this notation as "the user knows concept *c*". We do not wish to imply that there is no difference between the notions of "know" and "believe". In fact, we have chosen a notation that makes use only of the belief predicate BEL precisely because we do not wish to enter into a philosophical dispute about the differences between these two notions. In this work, we have not found the need to distinguish between "know" and "believe". A thorough treatment of the differences is beyond the scope of this work; see for example, Kobsa (1989).

[10] Note that this is a slight change from the way it is used in Kass (1991, p. 230), in which `(BEL USER (CONCEPT c))` is intended to mean that the user only knows of the *existence* of the concept *c*, but does not necessarily know any more information about it. In particular, in Kass (1991), this assertion is not intended to mean that the user knows the definition or any properties of c.

- (COMPETENT USER (DO USER *act*)): The system believes that the user is competent to perform the primitive domain action, *act*.
- (COMPETENT USER (ACHIEVE USER *goal*)): The system believes that the user is competent to achieve *goal*. That is, the user knows a method for achieving the non-primitive domain action.

In addition, we allow for the use of stereotypes. An individual user model can thus also include an assertion indicating a stereotype appropriate for the user (e.g., (ISA USER *stereotype*)).

3.4.2. Obtaining the user model

Our system builds individual user models from the following sources:

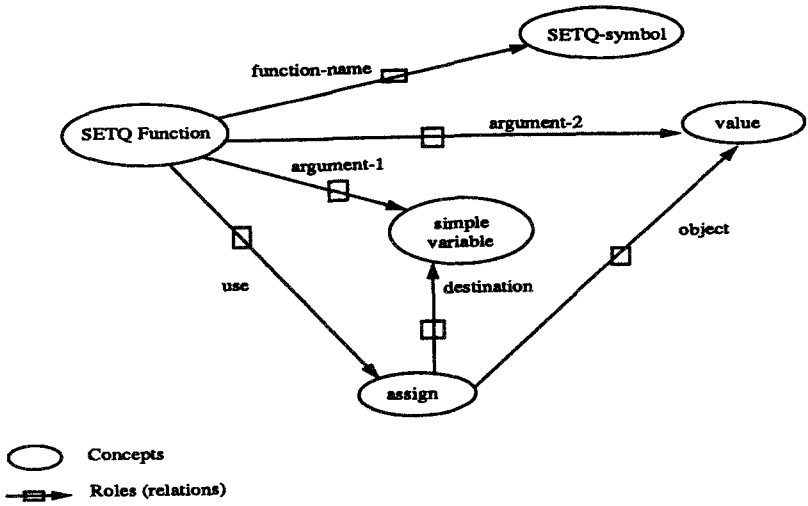
- a set of user stereotypes;
- the user's interactions with the system, including the initial request to the system and responses to the system's queries;
- observable artifacts, when available;

Obtaining information from user stereotypes. The system contains stereotypical user models, including models for system developers, domain experts, and novice users. Each stereotype includes some assertions that the system believes about such a stereotypic user. For example, the stereotype for a system developer in the domain of digital circuits indicates that the user knows the logical predicates EXISTS and FORALL and the concepts OBJECT-RELATION-ASCRIPTION, and OUTPUT-TERMINAL. On the other hand, the stereotype for naive users in that domain indicates that only the concept of OUTPUT (as opposed to OUTPUT-TERMINAL) is understood.

The system also contains a model of the "canonical user', which specifies a set of basic concepts that we presume to be known to all users of a given expert system. For example, in the domain of LISP programming, this set includes the concepts of PROGRAM, VARIABLE and FUNCTION. In the digital circuit domain, every user is presumed to know the concepts of CIRCUIT and CONNECTORS.

As in Rich (1989) and Chin (1989), an individual user model inherits information from the stereotypes that are applicable. Information from the stereotype is used when no other more specific information overrides it.

Obtaining information from interactions. The system can often obtain knowledge about the user's goals and beliefs from the interactive dialogue. First, since the user is employing the expert system, the system infers that the user shares its top-level goal. For example, the Program Enhancement Advisor (PEA) (Neches *et al.*, 1985) is an advice-giving system intended to aid



Paraphrase in English:

SETQ-FUNCTION has the function name, SETQ-SYMBOL, and its use is to ASSIGN a VALUE to a SIMPLE-VARIABLE. It has two arguments: a SIMPLE-VARIABLE, and a VALUE.

Fig. 4. PEA's knowledge about SETQ.

users in improving their Common LISP programs by recommending simple transformations that enhance the user's code. The top-level goal of this system is to enhance the program supplied by the user. The system can thus add the assertion (GOAL USER ENHANCE-PROGRAM) to the user model at the start of the interaction. Furthermore, whenever the system asks the user questions which explicitly request information about the user's goals, that information is recorded in the user model. For example, PEA begins its dialogue with the user by asking what characteristics of the program the user would like to enhance. Thus the system can update the user model to indicate that the user has the goal(s) of enhancing the chosen characteristic(s).

As the dialogue proceeds, the system can augment the user model. For example, if the system explains a concept to the user and no follow-up questions about this concept arise (or, if after a series of follow-up questions, the user indicates that the description is understood), the system can update the user model to indicate that the user knows the concept.

Obtaining information from observable artifacts. Finally, in some application domains, it is possible to gather information about the user from some artifact created by the user. In the PEA domain, for example, the system can glean some information about what LISP constructs the user knows from the program to be enhanced. To do this, before beginning the interaction with the user, the system makes one pass through the user's code to determine what functions the user has employed. For each function in the user's code, the system infers that the user knows the function, its use, and the concepts associated with that use. The user model is updated appropriately. Note that, because we use an overlay technique, the system can only make inferences about functions that appear in its domain model.

For example, suppose the user's program contains the S-expression `(SETQ X 1)`. PEA's knowledge about the `SETQ` function indicates that `SETQ` can be used to assign a value to a simple variable (as illustrated in Figure 4). As a result, the system infers that the user knows the concept `SETQ-FUNCTION`, as well as the concepts `ASSIGN`, `VALUE` and `SIMPLE-VARIABLE`.

Clearly this inference rule alone is too simplistic and may cause the system to attribute too much or too little to the user's knowledge state (e.g., if the user employed a function incorrectly). However, since our system allows the user to ask follow-up questions and to ask for elaboration if he or she is not satisfied with the system's explanations, such errors in the user model are not critical. This is indeed an advantage of a system that is capable of reacting to the user's feedback by providing elaborating and clarifying responses.

4. Exploiting the User Model During Text Generation

We now discuss the text planning mechanism in more detail, focusing on the role of the user model during the generation process. The user model guides the generation process at several stages. It affects plan selection through operator constraints that refer to the user model and through plan selection heuristics. In addition, the user model influences what and how much is ultimately included in the text. Finally, it is also used to interpret and handle the user's feedback. The next section provides a detailed example which illustrates how the system combines employment of the user model and feedback from the user to achieve effective communication.

4.1. CHECKING CONSTRAINTS AND MAKING ASSUMPTIONS

The planning process begins when a communicative goal is posted. The planner identifies all of the operators that are potentially capable of satisfying a given communicative goal by finding those whose effect field matches the

goal to be refined. For each operator found, the planner then checks to see if its constraints are satisfied.

As noted in Section 3.3, constraints may refer to facts about the domain or the problem-solving activity, as indicated in the expert system's knowledge bases, or to facts concerning the user, as recorded in the user model.[11] When the planner is checking the constraints on a plan operator, it attempts to find variable bindings in the expert system's knowledge bases or the user model that satisfy the constraints in the constraint list.

Two types of constraints may refer to the user model: one concerns specific beliefs or goals the user has, and the other concerns stereotypes that apply to the user. Constraints of the first type were illustrated in Figure 3 page 296, e.g., `(GOAL USER ?goal)`. A constraint of the second type is a query about whether the user belongs to a specific stereotype, e.g., `(ISA USER SYSTEM-DEVELOPER)`. This type of constraint is useful because certain explanation strategies are appropriate for some types of users and not others (Paris, 1988; Cohen and Jones, 1989). For example, the system may have two strategies available for justifying a conclusion: one which closely traces the reasoning of the expert system, and one that summarizes it by highlighting the main points. The first one is more appropriate for system developers (who are trying to debug the system and thus need to know precisely how the reasoning was done), while the second is more appropriate for end users (Paris, 1991). This preference is indicated in the constraints of the corresponding plan operators. In this way, the user model can trigger the use of specific strategies.

Constraints that refer to the expert system's knowledge are treated differently from constraints that refer to the user model. We assume that the information in the expert system's knowledge bases is correct. Therefore, constraints referring to aspects of this knowledge are treated as *rigid* constraints. If one of these constraints fails to be satisfied, the operator is rejected from consideration immediately. In contrast, since we do not wish to assume that our user model is either complete or correct, constraints referring to the user model are treated as a specification of what the user should know (what goals the user should have, or what type of user he or she should be) in order to understand the text that will be produced by the operator. When attempting to satisfy a constraint referring to the user model, the planner may make an assumption if the constraint is not satisfied according to the user model. We allow the planner to make assumptions because the user model could be incomplete.

[11] In general, constraints may also refer to information in the dialogue history or the evolving text plan. See Moore and Paris (forthcoming) for details.

For example, in checking the constraints of the plan operator in Figure 3 page 296, the system first checks the constraint `(STEP ?act ?goal)`. This constraint refers to an aspect of the expert system's problem-solving knowledge, and only bindings for the variable *?goal* satisfying this constraint will be considered acceptable. Suppose that, as will be the case in the example presented in Section 5, satisfying this constraint results in several possible bindings for the variable *?goal*. Some of these bindings will also satisfy the constraint that the goal be a goal of the user: `(GOAL USER ?goal)`. Others will not. Those bindings which do not satisfy this second constraint will not be rejected immediately. Instead, they will be noted as possible bindings, and each will be marked to indicate that, if this binding is used, the assumption that this is a goal of the user is being made.

Every possible binding environment for each operator is considered a candidate. Selection heuristics choose one operator, and a binding environment for that operator, from among the candidates. One of these heuristics can be set to indicate that the system should prefer operators that avoid making assumptions about the user's knowledge. When this is the case, the text planner prefers choosing binding environments that require no assumptions to be made. However, in any given situation, the influence of the other selection heuristics may outweigh this concern, as we will see in the next section. In that case, or if no assumption-free binding environment exists for an operator, one that requires assumptions will be chosen. Whenever the planner selects an operator, any assumptions that are made are recorded in the plan structure at the plan node where they occur.

Making assumptions at the grammar interface. There is another way that assumptions may arise during the planning process. This occurs when the planner reaches a primitive operator, i.e., a speech act such as `INFORM`. As we noted above, in order for the text generator to produce English text, the text planner must build a specification for each utterance in the input language required by the Penman sentence generator.

Until this point in the text planning process, the system has reasoned about concepts in the system's knowledge representation language. To generate text involving these concepts, references to, or descriptions of, these concepts must be planned. For example, consider the case where the system wishes to express the concept `ASSIGN-TO-GV`, which is represented in the knowledge base as follows:

```
ASSIGN-TO-GV = (ASSIGN (actor SETF-FUNCTION)
                       (object VALUE)
                       (destination GENERALIZED-VARIABLE))
```

During text planning, the system reasons only in terms of ASSIGN-TO-GV. This would be the case, for example, if this concept were part of a more complex predicate, such as (USE SETF-FUNCTION ASSIGN-TO-GV). However, to express the concept in English, it must now be "unpacked". An appropriate utterance expressing this concept might be: *"The* SETF *function assigns a value to a generalized variable."* To understand such an utterance, the hearer must know the concepts ASSIGN, VALUE, and GENERALIZED-VARIABLE. When building the SPL specification for an utterance, the grammar interface checks to see if the user knows the filler of each role to be expressed. If he or she does, the system can simply mention that concept by name in the generated text. If, on the other hand, the user model does not indicate that the user is familiar with the concept to be mentioned, an assumption is recorded at this point in the text plan.

For example, suppose that in planning how to express the concept ASSIGN-TO-GV, the user model indicates that the user knows the concepts ASSIGN and VALUE but has no indication that the user knows the concept GENERALIZED-VARIABLE. When the system plans the specification for the INFORM speech act involving ASSIGN-TO-GV, it will make an annotation at the appropriate node in the plan structure indicating that the system made the assumption that the user knew about GENERALIZED-VARIABLEs. If feedback from the user indicates that this utterance was not understood, the system uses information about such assumptions to determine how to recover from the misunderstanding.

Alternatives to making assumptions. Making assumptions is not the only possible recourse in these situations. One alternative is for the system to plan text explaining any unknown concepts. In fact, our system can operate in either of two modes: terse and verbose. In verbose mode, the system posts subgoals to describe any unknown concepts.

Yet another alternative is for the system to ask the user a question whenever it needs to use a concept that is not in the user model. In this way, the system could determine whether or not the concept is familiar to the listener. If so, planning could simply continue. If not, the system could engage in a subdialogue to explain that concept before continuing with the current explanation or, more drastically, could decide to abandon the current explanation strategy entirely, in favor of one that does not make use of concepts which are unfamiliar to the user.

4.2. SELECTING A STRATEGY

The planner employs *plan selection heuristics* based on several factors to choose an operator and a binding environment. These factors include what the user knows (as indicated in the user model), the conversation that has occurred so far (as indicated in the dialogue history), the relative specificity of the candidate operators, the assumption requirements of each operator (as indicated in the possible binding environments found when satisfying the constraints), and the amount of text an operator will generate.

Associated with each heuristic is a weight ranging from 0 to 1 indicating how that heuristic should contribute to the overall assessment of the plan operators. The magnitude of the weight indicates how much influence a particular heuristic should have on the decision. The polarity of the weight indicates whether that influence should affect the score positively or negatively. Positive weights indicate preference, while negative weights indicate avoidance. For example, setting the weight associated with the assumption heuristic to -0.9 can be paraphrased as "Strongly avoid operators that require making assumptions about the hearer's beliefs and goals." After each operator is given a score for each heuristic, the total score for each operator is set to the weighted sum of the individual scores. The candidate with the highest score is chosen.

The selection heuristics and the method for computing individual candidate scores are as follows:

- **Assumption Heuristic:** Avoid/Prefer operators that require making assumptions about the hearer's beliefs or goals. To compute the candidates' scores for this heuristic, the system counts the number of assumptions each candidate would require, as well as the maximum number of assumptions that any candidate in the candidate set would require. Each candidate is then assigned a score equal to the number of assumptions the operator requires divided by the maximum number of assumptions required by any operator in the set.
- **User Model Heuristic:** Prefer/Avoid operators that make use of concepts the hearer knows. To compute the score for a candidate, the system counts the number of concepts known to the user that appear in the nucleus and required satellite(s) of the candidate. This is an estimate of the number of familiar concepts that will be mentioned in the text if this candidate is chosen. Again, individual scores are normalized by dividing each score by the maximum score assigned to any candidate for this heuristic.

  Note that this heuristic differs from the assumption heuristic. The assumption heuristic is concerned with avoiding (or preferring!) concepts the user does *not* know. In contrast, the user model heuristic tries to

maximize (minimize) usage of concepts that the user *does* know. As an example, consider an operator that explains a concept, c_1, by drawing an analogy with concept c_2, which is known to the user. Because it makes no assumptions, this candidate receives a score of 0 from the assumption heuristic. Because it mentions c_2, it gets a score of $1/n$ from the user model heuristic (where n = the maximum number of known concepts used by any candidate). Another candidate operator may make 1 assumption and mention 1 known concept. All other things being equal, we would like the planner to choose the first candidate. It can only do this if we have two separate heuristics to keep track of these different concerns.

- **Coherence Heuristic:** Prefer/Avoid operators that make use of concepts previously mentioned in the dialogue history. The score for this heuristic is computed by counting the number of concepts that appear in the nucleus and satellite(s) of each operator that also appear in the dialogue history. The score is then normalized by dividing it by the maximum score assigned to any operator for this heuristic. Preferring the use of concepts previously mentioned in the dialogue history would result in a highly cohesive dialogue.
- **Specificity Heuristic:** Prefer/Avoid specific operators over more general ones. To compute scores, the operators are sorted from most general to most specific, based on the specificity of their constraints.[12] Each operator is given a score corresponding to its place in the sorted list. The score is then normalized by dividing by the total number of candidates in the list.
- **Verbosity Heuristic:** Prefer/Avoid operators that generate verbose responses. This is computed by counting the number of subgoals in the nucleus and required satellites, and normalizing the score by dividing it by the maximum score assigned to any operator.

The behavior of the explanation system can be modified by changing the weights on these heuristics. Although the weights are currently set by hand, we hope to devise a characterization of the types of explanations that are suitable for certain classes of users. This knowledge could then be encoded into a set of stereotypes that would determine a collection of weights appropriate to each stereotype. For example, it may be the case that for novice users the system should avoid making assumptions, should prefer operators that make use of concepts the user knows, and should avoid operators that generate verbose responses (the longer the text, the greater the chance of including concepts the

[12] See Moore (1989a) for details on how specificity of constraints is computed.

In Formal Notation:

EFFECT: **(GOAL ?hearer (DO ?hearer ?act))**
CONSTRAINTS: **(NUCLEUS)**
NUCLEUS:
(RECOMMEND ?speaker ?hearer ?act)
SATELLITES:
(((PERSUADED ?hearer
(GOAL ?hearer (DO ?hearer ?act))) *optional*)
((COMPETENT ?hearer (DO ?hearer ?act)) *optional*))

English Translation:

To make the hearer want to do an *act*,
IF this text span is to appear in nucleus position, THEN
1. Recommend the *act*
AND optionally,
2. Achieve state where the hearer is persuaded to do the *act*
3. Achieve state where the hearer is competent to do the *act*

Fig. 5. High-level plan operator for recommending an act (repeated).

user does not understand). In other circumstances, different settings would be appropriate.

4.3. OPERATOR EXPANSION

Once a plan operator has been selected, it is recorded in the plan node as the selected operator, and all other candidate plan operators are recorded as untried alternatives. The planner then instantiates the selected operator by posting its nucleus and required satellites as subgoals to be refined.

At this point, the planner must decide whether or not to expand optional satellites. As already mentioned, the planner currently has two modes, terse and verbose. In terse mode, no optional satellites are expanded. In verbose mode, the planner consults the user model and the dialogue history to determine whether or not each optional satellite subgoal should be expanded. If a satellite corresponds to a goal that has previously been achieved (as recorded in the dialogue history), or the user already has the knowledge or goal that this satellite would communicate (as indicated in the user model), then the satellite will not be expanded. Otherwise, it will be expanded.

For example, consider, the plan operator shown in Figure 5 which has

two optional satellites.[13] The first satellite calls for persuading the hearer to perform the act. The second satellite corresponds to making the hearer competent to perform the act. Now suppose that the planner is in verbose mode. If the user model indicates the user already has the goal of performing the act, then the first satellite will not be expanded. Similarly, if the system believes that the user already knows how to do the act being recommended, the second satellite will not be expanded.

5. Integrating a User Model and Feedback: An Extended Example

Here we illustrate how our system employs a user model to generate responses, and relies on feedback to compensate for inaccuracies in that model. Our example is taken from the Program Enhancement Advisor (PEA) (Neches *et al.*, 1985), which, as mentioned before, is an advice-giving system intended to aid users in improving their Common LISP programs by recommending transformations that enhance the user's code.[14] The user supplies PEA with a program and indicates which characteristics of the program should be enhanced (any combination of readability, maintainability, and efficiency). PEA then recommends transformations. After each recommendation is made, the user is free to ask questions about the recommendation.

Consider the sample dialogue shown in Figure 6. At the beginning of the interaction, PEA initializes the user model with information from the stereotypic canonical user model. In PEA, this set includes some very basic concepts about LISP programming and text editing that the system considers should be known to anyone using the PEA system. In particular, the system assumes that all users are capable of performing replacement acts, that they know what programs, functions, S-expressions and storage-locations are, and that they understand the characteristics that PEA can enhance, i.e., readability, maintainability, and efficiency. This canonical user model is shown in Figure 7. If the system knew of the applicability of other stereotypes, these would also be included in the user model.[15]

---

[13] The constraint of this operator, NUCLEUS, refers to the evolving text plan. This constraint is not explained here as it is not relevant to the discussion at hand. See Moore and Paris (forthcoming).

[14] PEA recommends transformations that improve the "style" of the user's code. It does not attempt to understand the content of the user's program.

[15] The system currently has no sophisticated way of triggering the use of a stereotype, such as by noticing that the user has employed certain terms (Chin, 1989; Chappel and Cahour, 1991). Stereotypes such as "system developers", "domain-expert", "domain-novice" must be specified at the outset of the interaction. We intend to use linguistic clues to trigger stereotypes in the future.

SYSTEM	What characteristics of the program would you like to enhance?	[1]
USER	Readability and maintainability.	[2]
	⋮	
SYSTEM	You should replace `(SETQ X 1)` with `(SETF X 1)`.	[3]
USER	Why?	[4]
SYSTEM	I am trying to enhance the maintainability of the program by applying transformations that enhance maintainability. A transformation that enhances maintainability is defined as a transformation whose right-hand side's use is more general than its left-hand side's use. `SETF` can be used to assign a value to any generalized variable. In contrast, `SETQ` can only be used to assign a value to a simple variable.	[5]
USER	Huh?	[6]
SYSTEM	Simple variables are storage locations named by symbols. This concept can be generalized to a storage location that can be named by any access function. These storage locations are called generalized variables.	[7]
USER	Huh?	[8]
SYSTEM	For example, the car of a cons is a generalized variable named by the access function `CAR` and the cdr of a cons is a generalized variable named by the access function `CDR`.	[9]

Fig. 6. Sample dialogue.

Next, the system examines the user's program to gather information about what LISP constructs the user knows. As was explained in Section 3.4.2, it does so by making one pass through the user's code to determine what functions have been employed, and for each function it recognizes, it records in the user model the function, its use, and any concepts related to this use. In this example, let us assume that this results in adding assertions indicating that the user knows the concepts: `CAR-FUNCTION`, `CDR-FUNCTION`, `SETQ-FUNCTION`, `CAR-OF-CONS`, `CDR-OF-CONS`, `SIMPLE-VARIABLE`, `ASSIGN` and `VALUE`.

The system then begins the dialogue by asking what characteristics of the program the user would like to enhance. When the user responds with a choice of characteristics, the information that the user has the goal(s) of enhancing those characteristics is added to the user model. In this case, the following two entries are added to the user model:

```
(GOAL USER ENHANCE-READABILITY)
(GOAL USER ENHANCE-MAINTAINABILITY)
```

```
(COMPETENT USER (DO USER REPLACE))
     ; The user is capable of performing replacement acts
(BEL USER (CONCEPT PROGRAM))
     ; The user knows the concept of a program
(BEL USER (CONCEPT LISP-FUNCTION))
     ; The user knows the concept of a LISP function
(BEL USER (CONCEPT STORAGE-LOCATION))
     ; The user knows the concept of storage location
(BEL USER (CONCEPT S-EXPR))
     ; The user knows the concept of an S-expression
(BEL USER (CONCEPT SYMBOL))
     ; The user knows the concept of a symbol
(BEL USER (CONCEPT ACCESS-FUNCTION))
     ; The user knows the concept of an access-function
(BEL USER (CONCEPT READABILITY))
     ; The user knows the concept of readability
(BEL USER (CONCEPT MAINTAINABILITY))
     ; The user knows the concept of maintainability
(BEL USER (CONCEPT EFFICIENCY))
     ; The user knows the concept of efficiency
```

Fig. 7. Canonical user model in PEA.

The resulting user model is shown in Figure 8.

The system can now begin recommending transformations to achieve the user's goals. To make a recommendation, the expert system posts a communicative goal to the text planner. In the current example, the following communicative goal is posted: `(GOAL USER (DO USER REPLACE-1))`, where `REPLACE-1` corresponds to replacing `(SETQ X 1)` with `(SETF X 1)`. The plan operator of Figure 5 page 310 is chosen. In this example, the system is in terse mode. The optional satellites are thus not expanded. (Note that the second optional satellite would not be expanded even in verbose mode since the user model indicates that the user is competent to perform replacement acts.) Thus only the nucleus of the operator is expanded. This is a speech act, so planning is complete and the system recommends that the user replace `(SETQ x 1)` with `(SETF x 1)` in turn 3 of the sample dialogue.

The user responds by asking "Why?", turn 4, thus indicating that he or she is not immediately convinced that this replacement should be done and wants the system to justify this recommendation. As a result of this question, the query analyzer posts the communicative goal to achieve the state in which

```
(GOAL USER ENHANCE-READABILITY)
(GOAL USER ENHANCE-MAINTAINABILITY)
(BEL USER (CONCEPT CAR-FUNCTION))
(BEL USER (CONCEPT CDR-FUNCTION))
(BEL USER (CONCEPT SETQ-FUNCTION))
(BEL USER (CONCEPT CAR-OF-CONS))
(BEL USER (CONCEPT CDR-OF-CONS))
(BEL USER (CONCEPT SIMPLE-VARIABLE))
(BEL USER (CONCEPT ASSIGN))
(BEL USER (CONCEPT VALUE))
```

The user model also inherits the following assertions from the canonical user model:

```
(COMPETENT USER (DO USER REPLACE))
(BEL USER (CONCEPT PROGRAM))
(BEL USER (CONCEPT LISP-FUNCTION))
(BEL USER (CONCEPT STORAGE-LOCATION))
(BEL USER (CONCEPT S-EXPR))
(BEL USER (CONCEPT SYMBOL))
(BEL USER (CONCEPT ACCESS-FUNCTION))
(BEL USER (CONCEPT READABILITY))
(BEL USER (CONCEPT MAINTAINABILITY))
(BEL USER (CONCEPT EFFICIENCY))
```

Fig. 8. Contents of user model in example.

user is persuaded to do the recommended act, i.e, `(PERSUADED USER (GOAL USER (DO USER REPLACE-1)))`. One of the strategies for persuading the listener to perform an act is to find goals that this act is a step in achieving and to motivate the act in terms of these goals. The plan operator that embodies this strategy was shown in Figure 3 on page 296. This operator is shown again in Figure 9 for convenience.

When attempting to satisfy the constraints of this operator, the system first checks the constraint `(STEP REPLACE-1 ?goal)`. This constraint states that, in order to use this operator, the system must find a domain goal, *?goal*, that `REPLACE-1` is a step in achieving. To find such goals, the planner searches the expert system's development history for goals that led to recommending `REPLACE-1`. In this example, the applicable expert system goals, listed in

In Formal Notation:

EFFECT: **(PERSUADED ?hearer (GOAL ?hearer (DO ?hearer ?act)))**
CONSTRAINTS: **(AND (STEP ?act ?goal)**
(GOAL ?hearer ?goal))
NUCLEUS: **(FORALL ?goal**
(MOTIVATION ?act ?goal))
SATELLITES: **nil**

English Translation:

To achieve the state in which the hearer is persuaded to do an *act*,
IF the *act* is a step in achieving some *goal(s)* of the hearer,
THEN motivate the *act* in terms of those *goal(s)*.

Fig. 9. Plan operator for persuading user to do an act.

order from most to least specific, are:

```
APPLY-SETQ-TO-SETF-TRANSFORMATION
APPLY-LOCAL-TRANSFORMATIONS-WHOSE-RHS-
        USE-IS-MORE-GENERAL-THAN-LHS-USE
APPLY-LOCAL-TRANSFORMATIONS-THAT-ENHANCE-MAINTAINABILITY
APPLY-TRANSFORMATIONS-THAT-ENHANCE-MAINTAINABILITY
ENHANCE-MAINTAINABILITY
ENHANCE-PROGRAM
```

Thus, six possible bindings for the variable *?goal* result from the search for domain goals that `REPLACE-1` is a step in achieving.

The second constraint of the current plan operator, `(GOAL ?hearer ?goal))`, is a constraint on the user model stating that *?goal* must be a goal of the hearer. Not all of the bindings found so far will satisfy this constraint. As noted before, those which do not will not be rejected immediately, as we do not assume that the user model is complete. Instead, they will be noted as possible bindings, and each will be marked to indicate that, if this binding is used, an assumption is being made, namely that the binding of *?goal* is assumed to be a goal of the user.

In this example, since the user is employing the system to enhance a program and has indicated a desire to enhance the readability and maintainability of the program, the system has inferred the user shares the top-level goal of the system `ENHANCE-PROGRAM`, as well as the two more specific goals

ENHANCE-READABILITY and ENHANCE-MAINTAINABILITY. Therefore, the two goals that completely satisfy the first two constraints of the operator shown in Figure 9 are ENHANCE-PROGRAM and ENHANCE-MAINTAINABILITY. But note that ENHANCE-MAINTAINABILITY is a refinement of ENHANCE-PROGRAM. In order to avoid explaining parts of the reasoning chain that the user is familiar with the most specific goal is chosen, when one goal is a subgoal of another. Therefore, ENHANCE-MAINTAINABILITY is now the preferred candidate binding for the variable *?goal.*

The plan operator is thus instantiated with ENHANCE-MAINTAINABILITY as the binding for the variable *?goal.* The operator is recorded in the plan node as the selected operator, and all other candidate operators are recorded as untried alternatives.

The nucleus of this operator is now expanded. Note that the nucleus contains the special form FORALL. In general, the FORALL form causes the planner to create a subgoal for each of the possible bindings of the variable that FORALL ranges over. In this case, since there is only one such binding, the single subgoal

```
(MOTIVATION REPLACE-1 ENHANCE-READABILITY)
```

is posted.

One strategy for satisfying this type of motivation goal is to inform the hearer of the goal that the system is trying to achieve and then to establish that the act in question is part of the means for achieving the goal. The plan operator for this strategy is shown in Figure 10.

This operator is selected and planning continues in this fashion until all subgoals have been refined to speech acts. The completed text plan, shown in Figure 11, is recorded in the dialogue history, and the system's utterance in turn 5 of the dialogue is generated:

SYSTEM I am trying to enhance the maintainability of the program by applying transformations that enhance maintainability. A transformation that enhances maintainability is defined as a transformation whose right-hand side's use is more general than its left-hand side's use. SETF can be used to assign a value to any generalized variable. SETQ can only be used to assign a value to a simple variable.

Note that this strategy first states the goal that the system is trying to achieve, and then explains how the recommended act is part of achieving this goal. In this case, the system is enhancing maintainability by applying

EFFECT: **(MOTIVATION ?act ?goal)**
CONSTRAINTS: **(AND (STEP ?act ?goal)**
(GOAL ?hearer ?goal))
NUCLEUS: **(INFORM ?speaker ?hearer ?goal))**
SATELLITES: **(((MEANS ?goal ?act) *required*))**

English Translation:

To motivate a *goal* in terms of an *act*,
IF the *goal* is a goal of the hearer,
and the *act* is a step towards achieving the *goal*
THEN
Inform the hearer of the *goal*
AND explain the means by which this *goal* is achieved.

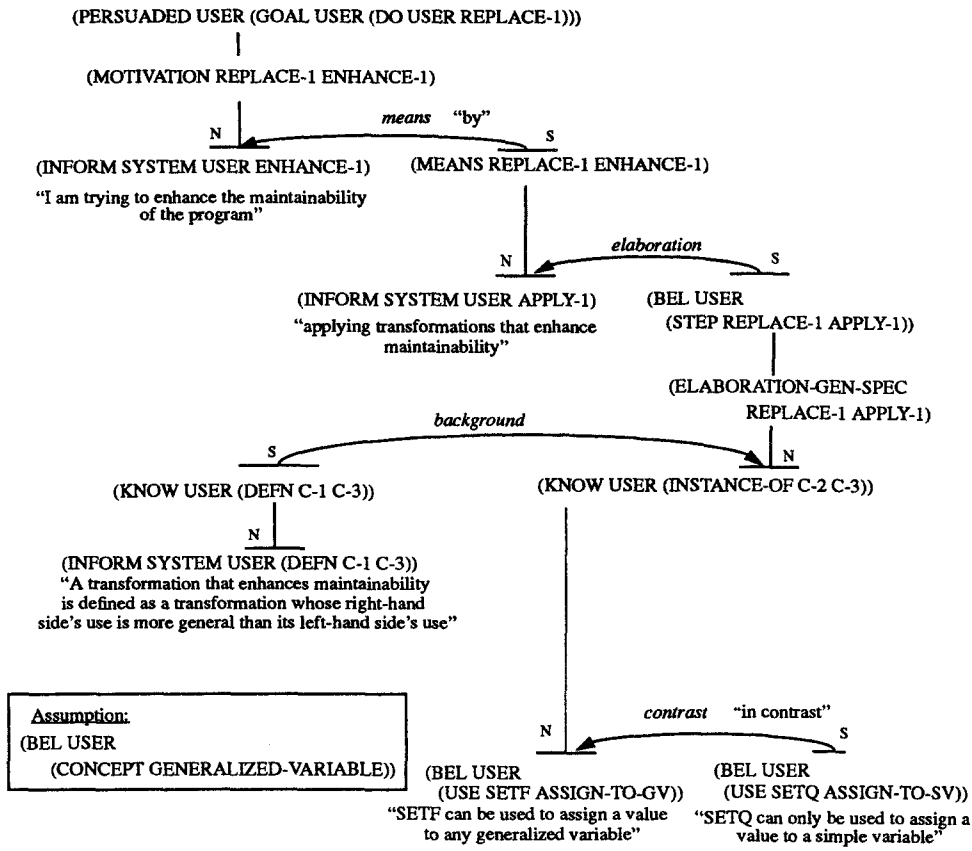
Fig. 10. Plan operator for motivating any action by stating how a shared goal is achieved.

transformations that enhance maintainability. Replacing SETQ with SETF is an instance of a transformation that enhances maintainability. The explanation gives a definition of the concept "transformations that enhance maintainability", and then an explanation of why the instance fits this definition.

To generate this response, the system builds a definition of the concept from information in its domain model. It then explains how SETQ-TO-SETF fits the definition, i.e., SETF has more general usage than SETQ, by stating the respective uses of SETQ and SETF. To present the use of SETF, the speech act `(INFORM SYSTEM USER (USE SETF-FUNCTION ASSIGN-TO-GV))`, must be uttered. To express this speech act, the system needs to expand the concept ASSIGN-TO-GV:

```
ASSIGN-TO-GV = (ASSIGN (actor SETF-FUNCTION)
                       (object VALUE)
                       (destination GENERALIZED-VARIABLE))
```

Recall from Section 4.1, that the SPL specification for this process includes the concepts ASSIGN, VALUE and GENERALIZED-VARIABLE. The system must now decide whether lexical items attached to each concept ("assign", "value", and "generalized-variable") can be used to express each concept, or whether it needs to plan a definition for one or more of them. This is in part determined by looking at the user model to check whether the user knows the concepts involved. If he or she does, then the concept is referred to by the lexical item attached to the concept. Otherwise, the behavior of the system will depend



RST relation connecting the two text spans
N: Nucleus of the RST relation
S: Satellite of the RST relation

REPLACE-1 = Replace (SETQ X 1) with (SETF X 1)
ENHANCE-1 = Enhance maintainability
APPLY-1 = Applying transformations that enhance maintainability
C-1 = Transformations that enhance maintainabillity
C-2 = SETQ-to-SETF transformation
C-3 = Transformation whose right-hand side's use is more general than its left-hand side's use
ASSIGN-TO-GV = Assign a value to a generalized variable
ASSIGN-TO-SV = Assign a value to a simple variable

Fig. 11. Completed text plan for utterance [5] of the sample dialogue.

on its current mode. In terse mode, the planner makes an assumption that the user knows the concept, and awaits feedback from the user about the success of the generated text. In verbose mode, a communicative goal to define the object would be posted to the text planner, and a longer text including the definition would immediately be generated.

In our example, the system is in terse mode, and the user model indicates that the user knows the concepts ASSIGN and VALUE but has no indication that the user knows the concept GENERALIZED-VARIABLE. (The contents of the user model were shown in Figure 8 page 314.) Thus, when planning the SPL specification for the INFORM involving ASSIGN-TO-GV, the system makes an annotation at that node in the plan structure indicating that the system made the assumption that the user knows the concept GENERALIZED-VARIABLE. (See Figure 11). This is the case because the user model could be incomplete.

In this example, the user responds with the vaguely-articulated question, "Huh?" on line 6. The system thus examines the text plan that produced the previous response (turn 5) to determine what could have gone wrong. There it finds a record that it made the assumption that the concept GENERALIZED-VARIABLE was known to the user. This assumption is now questioned, and the query analyzer posts a goal to satisfy it. This demonstrates that keeping track of the assumptions that were made in planning a text aids in allowing a system to elaborate even when the user cannot formulate a precise follow-up question.

Now consider what would happen if the user model incorrectly indicated that the user knew the concept GENERALIZED-VARIABLE. In that case, no assumption would be recorded in the text plan, and, therefore, the system could not determine that this concept might be causing the misunderstanding. In this case, the recovery mechanism would conclude that the top-level goal failed and would try to replan the entire explanation. If no other strategies were available, the system would present the user with a menu of follow-up questions that it can answer about its prior explanation. This menu would include "What is a generalized variable?".

Continuing with the example, the system must now construct a plan for achieving the goal (BEL USER (CONCEPT GENERALIZED-VARIABLE)). In its library of plan operators, the system has several plan operators for making the hearer know about concepts. It may describe a concept by stating the concept's attributes and its parts, by drawing an analogy with a similar concept, by giving examples of the concept, by generalizing a concept the user is familiar with, or by identifying the object as a member of a superclass and explaining the essential differences between the object and the superclass.

In this case, the three plan operators shown in Figure 12 are ap-

plicable: DESCRIBE-BY-SUPERCLASS, DESCRIBE-BY-ABSTRACTION, and DESCRIBE-BY-EXAMPLE. The other strategies for describing a concept, e.g., DESCRIBE-BY-ANALOGY and DESCRIBE-BY-PARTS-AND-USE, are not applicable because some of their constraints are not satisfied. For example, the strategy DESCRIBE-BY-ANALOGY is not a candidate operator because the knowledge base does not contain a concept that is a sibling of GENERALIZED-VARIABLE.

To choose from among these candidate plan operators, the planner employs the selection heuristics described in Section 4.2. We discuss only the heuristics that concern the user model, and assume that the scores for the remaining heuristics are the same for the three operators.

None of these operators require an assumption about the user to be made, as the user model indicates that the user knows the concepts STORAGE-LOCATION (the super-concept used in DESCRIBE-BY-SUPERCLASS), SIMPLE-VARIABLE (the sub-concept used in DESCRIBE-BY-ABSTRACTION), and CAR-OF-CONS and CDR-OF-CONS, (the two possible bindings for *?example* in DESCRIBE-BY-EXAMPLE). So the score for the assumption heuristic for all of these operators is 0.

The user model heuristic varies, however, as the second operator, DESCRIBE-BY-ABSTRACTION, would present text that makes a connection with four elements of the user's knowledge, (STORAGE-LOCATION, SIMPLE-VARIABLE, SYMBOL and ACCESS-FUNCTION), while the other two operators would only relate to two concepts the user already knows (STORAGE-LOCATION and ACCESS-FUNCTION for DESCRIBE-BY-SUPERCLASS, and CAR-OF-CONS and CDR-OF-CONS, for DESCRIBE-BY-EXAMPLE). With all other factors equal, the operator DESCRIBE-BY-ABSTRACTION is thus ranked highest by the plan selection heuristics, and it is chosen. The final text produced by employing this operator first describes SIMPLE-VARIABLEs and then abstracts this concept to introduce GENERALIZED-VARIABLEs (turn 7).

The user then indicates that he or she still does not understand this description ("Huh?", on line 8). To handle this query, the query analyzer again begins by examining the text plan that produced the system's previous response to determine which goal might have failed. It determines that, in this case, it is the top-level goal, (BEL USER (CONCEPT GENERALIZED-VARIABLE)), that failed and must therefore be replanned. This time, when the planner selects an operator for achieving this goal, it uses a set of recovery heuristics to determine which of the untried alternatives should be tried next.[16] One of the

[16] See Moore (1989a) for a detailed discussion of both the heuristics for determining which goal may have failed and the recovery heuristics and their application in handling "Huh?" questions.

Operator 1:

NAME: **describe-by-superclass**
EFFECT: **(BEL ?hearer (CONCEPT ?concept))**
CONSTRAINTS:
(AND (SUBCLASS ?concept ?super-concept)
(BEL ?hearer (CONCEPT ?super-concept))
(DIFFERENCE ?diff ?concept ?super-concept))
NUCLEUS: **(INFORM ?speaker ?hearer (CLASS-ASCRIPTION ?concept ?super-concept))**
SATELLITES: **(((FORALL ?diff (ELABORATE-OBJECT-ATTRIBUTE ?concept ?diff))))**

Score for the heuristics related to the user model:
assumption-score: 0 ; No assumptions need to be made
user-model-score: 2/4 ; The text will relate to 2 already known concepts

Operator 2:

NAME: **describe-by-abstraction**
EFFECT: **(BEL ?hearer (CONCEPT ?concept))**
CONSTRAINTS: **(AND (SUBCLASS ?sub-concept ?concept)**
(BEL ?hearer (CONCEPT ?sub-concept))
(IMMEDIATE-SUBCLASS ?concept ?super-concept))
NUCLEUS:
((SETQ ?diffs (FIND-ESSENTIAL-DIFFERENCES ?sub-concept ?super-concept))
(SETQ ?subc-attrs (GET-ATTRS-OF-CONCEPT-FROM-DIFFS ?sub-concept ?diffs))
(SETQ ?concept-attrs (GET-ATTRS-OF-CONCEPT-FROM-DIFFS ?concept ?diffs))
(INFORM ?speaker ?hearer
(CLASS-ASCRIPTION ?sub-concept (?super-concept ?subc-attrs))))
SATELLITES: **(((ABSTRACTION ?sub-concept ?concept ?super-concept ?concept-attrs)))**

Score for the heuristics related to the user model:
assumption-score: 0 ; No assumptions need to be made
user-model-score: 4/4 ; The text will relate to 4 already known concepts

Operator 3:

NAME: **describe-by-example**
EFFECT: **(BEL ?hearer (CONCEPT ?concept))**
CONSTRAINTS: **(AND (INSTANCE-OF ?example ?concept))**
(BEL ?hearer (CONCEPT ?example))
NUCLEUS: **(((FORALL ?example**
(ELABORATE-CONCEPT-EXAMPLE ?concept ?example)))
SATELLITES: **nil**

Score for the heuristics related to the user model:
assumption-score: 0 ; No assumptions need to be made
user-model-score: 2/4 ; The text will relate to 2 already known concepts

Fig. 12. Three plan operators for describing a concept.

recovery heuristics says that if the goal to be replanned is of the form (BEL USER (CONCEPT ?concept)), then the strategy of giving examples should be tried next, if applicable.

In this case, as indicated in the recorded text plan, the list of untried alternatives for describing a concept are:

DESCRIBE-BY-SUPERCLASS
DESCRIBE-BY-EXAMPLE

Thus, there are examples of the concept GENERALIZED-VARIABLE that are familiar to the user. Since DESCRIBE-BY-EXAMPLE is on the list of applicable alternatives, it will be chosen. This plan operator was shown as the third operator in Figure 12. To be applicable, this plan operator requires that there be at least one concept, *?example*, that is an immediate subclass of GENERALIZED-VARIABLE and that is familiar to the user. PEA's knowledge base contains several concepts that are immediate subclasses of GENERALIZED-VARIABLE, including CAR-OF-CONS, CDR-OF-CONS, CADR-OF-CONS, etc. In this example, the user model indicates that the user is familiar with the concepts CAR-OF-CONS and CDR-OF-CONS. Using these two concepts as bindings for the variable *?example*,[17] the system generates the recovery response on line 9 of the sample dialogue.

In this example, we have shown how our system is able to interpret and answer a user's follow-up question in context. The system was able to generate an explanation even in the absence of a complete user model, and, by recording the assumptions that were made, the system was able to respond intelligently to an indication that its explanation was not understood. Furthermore, we have seen that if the user indicates that an explanation was not understood and no assumptions were made, the system chooses an alternative strategy for producing a clarifying explanation.

6. Related Work

Many systems have employed a user model to improve the quality of the texts they generate (e.g., Appelt, 1985; McCoy, 1985; van Beek, 1986; Paris, 1988; Wolz *et al.*, 1990). Others are capable of making inferences about the user's current goals and plans (e.g., McKeown, 1988; Pollack, 1986b; Carberry, 1988), or about the user's preferences (e.g., Hoeppner *et al.*, 1984; Morik and Rollinger, 1985) in order to provide relevant advice to the user. These systems reflect the emphasis of the research at the time: how can a system exploit a

[17] Currently, all the possible bindings for the variable *example* would be expressed in the text. This is clearly inappropriate if there are many such examples. On-going work is addressing exactly these issues (see Mittal and Paris, 1992).

user model to improve its behavior, and what needs to be included in such a model to be useful. The focus of these efforts is on how to take advantage of a user model to improve the system's responses. These systems were thus attempting to generate the "best" answer in one shot, given an appropriate user model.

It is only recently that there has been a realization of the importance of dialogue capabilities in explanation systems, where the content of the explanation may be negotiated, and clarifications or elaborations of uncertain points sought. This concern is reflected in current research efforts, each taking a different approach to the problem of allowing dialogues. In Hartley and Smith (1988), a menu of follow-up questions is provided after an explanation is presented. In the MMI2 project, the system records the dialogue mainly to interpret anaphora in the user's queries (Chappel and Cahour, 1991). The system also handles clarification subdialogues, but of a different nature than the ones we handle here. In MMI2, subdialogues are initiated by the system when it needs to obtain more information from the user in order to perform its problem-solving activity.

In addition, Cawsey's EDGE system allows for interruptions from the user while a text is being generated (Cawsey, in press). EDGE plans tutorial explanations about the structure and input/output behavior of simple electrical circuits. EDGE plans an extended explanation at a high-level, following a specified curriculum. This plan is fleshed out as the dialogue progresses, causing sentences to be generated. After each sentence is generated, the system pauses to allow feedback from the user. The user supplies feedback by choosing an item from a menu. Because of the representation of the curriculum, i.e., a description of the topics that are to be covered in the explanation, if the user asks about a topic that is planned to be addressed later, EDGE can make a comment such as "We'll be getting to that in a moment." If this is not the case, or if the user insists, EDGE answers the user's question immediately. Once the interruption has been addressed, EDGE proceeds with its explanation as specified in the overall explanation plan. In essence, Cawsey's system takes a more global view of the dialogue than does ours because it plans an extended explanation. To handle interruptions from the user, EDGE does not need to reason about its own previous utterances. It only needs to reason about the high-level goals (the topics) it is trying to accomplish. This approach is particularly appropriate in a tutoring discourse where the system has a notion of an overall curriculum it wishes to present. However, in expert system applications, the system presents the user with a recommendation or result and only provides explanations when the user requests them. It is not appropriate for the system to plan extended explanations, testing the user's understanding,

and elaborating without provocation. Rather the structure of the dialogue only emerges as the user asks questions. Thus a system such as ours must be able to reason about its previous utterances on demand. As a result, our system must use the dialogue history in more sophisticated ways, i.e., in selecting plans to achieve discourse goals and interpreting and responding to users' questions in context.

It is clear that these methods are complementary, and that a complete system would need to incorporate them all. In particular, it would be interesting to merge Cawsey's technique with ours to obtain a more flexible system. A system might indeed need to both plan extended explanations and yet allow for clarification subdialogues initiated by the user to take place, as allowed in our system.

7. Future Work

There are still numerous issues to be addressed. We outline here the most important ones.

7.1. Building and Updating the User Model

As we explained here, our system uses a few simple heuristics to gather information about the user. We are investigating the incorporation of the more sophisticated inference rules proposed by Kass (1991) for acquiring a user model from an advisory dialogue.

In addition, our current system updates the user model only by adding information from the user's responses to the system's questions. Clearly, the system could make more sophisticated updates if it had a model of how the user's follow-up behavior should affect the user model. For example, if the system generates a definition of a concept, and the user does not ask a follow-up question immediately after the definition has been presented, the system could update the user model to indicate that the user now knows this concept. We believe that empirical studies are needed to determine what a system should conclude from users' follow-up behavior in advisory dialogues. However, because our system maintains a dialogue history that contains the user's questions as well as the text plans that produced the system's responses, we believe that our approach provides the framework for supporting the kind of reasoning about prior explanations that a more sophisticated updating scheme would require.[18]

[18] Furthermore, the issue of what, if any, are the differences between a dialogue history and a user model and when things should migrate from the dialogue history to the user model is still an open question. See Schuster *et al.* (1988) for the opinions of several researchers on this subject.

7.2. PLAN SELECTION

In the current implementation, the preferences (or weights) that affect the plan selection heuristics are set by hand and they remain constant throughout the dialogue. Clearly there are many research issues to be studied here. For example, rather than setting the weights by hand, the system could allow the user to express pragmatic goals such as "be brief/verbose", "tie in explanations with things the user already knows", "give general explanations" and so forth. The system would then require a set of rules to determine how these pragmatic goals should be translated into an appropriate set of weights for the selection heuristics. Alternatively, the explainer could alter the weights automatically as the dialogue progresses if the user's feedback indicates that the current weights are inappropriate. Or the system may be able to make use of the information in the user model to set the weights. For example, just as certain types of explanations are suitable for certain classes of users (Paris, 1988, 1991), particular sets of preferences might be suitable for some classes of users. This knowledge could be encoded into a set of stereotypes that would determine a collection of weights appropriate to each stereotype.

8. Concluding Remarks

A user model can be used to guide the generation process in providing explanations that are appropriate to users. However, it is not feasible to provide systems with complete and correct user models. It is thus important that a system not be critically dependent on the quality of its user model in order to provide adequate responses to its users. To communicate effectively, systems must be able to accept and handle feedback from users. In particular, they must be able to clarify misunderstood explanations, elaborate on previous explanations, and respond to follow-up questions in the context of an on-going dialogue.

We have described an explanation system that combines the capabilities of employing information in a user model when it is available and employing feedback from the user. Our explanation generation facility plans explanations from a rich set of strategies, keeping track of the system's discourse goals, the plans used to achieve them, and any assumptions made while planning a response. Our system maintains a recorded history of the text plans used in producing responses so that it can later reason about its own responses when feedback from the user indicates that an explanation was not satisfactory. Thus our system does not rely on the user model alone, but, instead, reacts to feedback and recovers from communication failure when it occurs.

In addition to enabling effective communication, the ability to take feed-

back into consideration can greatly facilitate the acquisition and maintenance of user models. It allows the system to discover additional information about the user and also makes it possible to identify cases where the user model contains erroneous information. Moreover, while the user model is being constructed, the system must be able to produce explanations with an incomplete and possibly incorrect user model. Thus we believe that systems having the capabilities both to exploit a user model and to react to feedback when that model proves insufficient show the most promise.

Acknowledgments

The research described in this paper was supported in part by the Defense Advanced Research Projects Agency (DARPA) under a NASA Ames cooperative agreement number NCC 2-520. Johanna Moore is currently supported by grants from the Office of Naval Research Cognitive and Neural Sciences Division, the National Science Foundation, and the National Library of Medicine. Cécile Paris gratefully acknowledges the support of DARPA under the contract DABT63-91-C-0025 while writing this paper.

The authors would like to thank William Swartout, who has advised us on this work since its inception. We would also like to thank Vibhu Mittal, Béatrice Cahour, and the anonymous reviewers who provided useful comments on earlier drafts of this paper.

References

Appelt, D. E.: 1985, *Planning English Sentences*. England: Cambridge University Press, Cambridge.

Bateman, J. A. and C. L. Paris: 1989, 'Phrasing a Text in Terms the User Can Understand'. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 20–25, pp. 1511–1517.

Bateman, J. A. and C. L. Paris: 1991, 'Constraining the Deployment of Lexicogrammatical Resources During Text Generation: Towards a Computational Instantiation of Register Theory'. In: Eija Ventola (ed.): *Functional and Systemic Linguistics: Approaches and Uses*. Mouton de Gruyter: Chapter 5, pp. 81–106.

Brown, J. S. and R. R. Burton: 1978, 'Diagnostic Models for Procedural Bugs in Basic Mathematical Skills'. *Cognitive Science* 2(2), 155–192.

Buchanan, B. G. and E. H. Shortliffe: 1984, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison-Wesley Publishing Company.

Bunt, H. C.: 1990, 'Modular Incremental Modelling of Belief and Intention'. In *Proceedings of the Second International Workshop on User Modeling*.

Cahour, B.: 1990, 'Competence Modelling in Consultation Dialogs'. In: L. Berlinguet and D. Berthelette (eds.): *Proceedings of the International Congress, Work With Dispay Units' 89*, Montreal, Canada, September 1990. Amsterdam: North-Holland.

Cahour, B.: 1991, *La Modélisation de l'Interlocuteur: Elaboration du Modèle et Effets au Cours de Dialogues de Consultation.* PhD thesis, Université Paris 8, France.

Carberry, S. M.: 1988, 'Modeling the User's Plans and Goals'. *Computational Linguistics* **14**(3), 23–37.

Carbonell, J. R.: 1970, 'AI in CAI: An Artificial Intelligence Approach to Computer-Aided Instruction'. *IEEE Transactions on Man-Machine Systems* **11**, 190–202.

Carr, B. and I. Goldstein: 1977, 'Overlays: A Theory of Modelling for Computed Aided Instruction'. AI Memo 406.

Cawsey, A.: in press, 'Planning Interactive Explanations'. *International Journal of Man-Machine Studies.*

Chandrasekaran, B. and W. Swartout: 1991, 'Explanations in Knowledge Systems: The Role of Explicit Representation of Design Knowledge'. *IEEE Expert* **6**(3), 47–50.

Chappel, H. and B. Cahour: 1991, 'User Modeling for Multi-Modal Co-Operative Dialogue with KBS'. Deliverable D3, Esprit Project P2474.

Chin, D. N.: 1987, *Intelligent Agents as a Basis for Natural Language Interfaces.* PhD thesis, University of California at Berkeley.

Chin, D. N.: 1989, 'KNOME: Modeling What the User Knows in UC'. In: A. Kobsa and W. Wahlster (eds.): *User Models in Dialog Systems.* Symbolic Computation Series. Berlin, Heidelberg, New York, Tokyo: Springer-Verlag.

Cohen, R. and M. Jones: 1989, 'Incorporating User Models into Expert Systems for Educational Diagnosis'. In: A. Kobsa and W. Wahlster (eds.): *User Models in Dialog Systems*, Symbolic Computation Series. Berlin, Heidelberg, New York, Tokyo: Springer-Verlag, pp. 35–51.

Falzon, P.: 1987, 'Les Dialogues de Diagnostic: L'évaluation des Connaissances de l'Interlocuteur'. Technical Report 747, INRIA, Rocquencourt, France.

Finin, T. W., A. K. Joshi, and B. L. Webber: 1986, 'Natural Language Interactions with Artificial Experts'. *Proceedings of the IEEE* **74**(7), July.

Haimowitz, I.: 1990, 'Modeling All Dialogue System Participants to Generate Empathetic Responses'. In *Proceedings of the Second International Workshop on User Modeling*, Honolulu, HI.

Hartley, R. and M. Smith: 1988, 'Question-Answering and Explanation Giving in On-Line Help Systems'. In: J. Self (ed.): *Artificial Intelligence and Human Learning*, Chapman and Hall, pp. 338–360.

Hoeppner, W., K. Morik, and H. Marburger: 1984, 'Talking it Over: The Natural Dialog System HAM-ANS'. Technical Report ANS-26, Research Unit for Information Science and Artificial Intelligence, University of Hamburg.

Jameson, A. and W. Wahlster: 1982, 'User Modelling in Anaphora Generation: Ellipsis and Definite Description'. In *Proceedings of 82 European Conference on Artificial Intelligence*, pp. 222–227.

Joshi, A., B. Webber, and R. Weischedel: 1984, 'Living Up to Expectations: Computing Expert Responses'. In *Proceedings of AAAI-84*, pp. 169–175. American Association of Artificial Intelligence.

Kasper, R. T.: 1989, 'A Flexible Interface for Linking Applications to Penman's Sentence Generator'. In *Proceedings of the Darpa Workshop on Speech and Natural Language.*

Kass, R.: 1991, 'Building a User Model'. *User Modeling and User-Adapted Interaction* **1**(3), 203–258.

Kobsa, A.: 1984, 'Generating a User Model from WH-Questions in the VIE-LANG System'. Technical Report 84-03, Department of Medical Cybernetics, University of Vienna.

Kobsa, A.: 1989, 'A Taxonomy of Beliefs and Goals for User Models in Dialog Systems'. In: A. Kobsa and W. Wahlster (eds.): *User Models in Dialog Systems.* Symbolic Computation Series. Berlin, Heidelberg, New York, Tokyo: Springer-Verlag.

Lehman, J. F. and J. G. Carbonell: 1989, 'Learning the User's Language: A Step Towards Automated Creation of User Models'. In: A. Kobsa and W. Wahlster (eds.): *User Models*

in Dialog Systems. Berlin, New York: Springer-Verlag.

Mann, W. C. and C. M. I. M. Matthiessen: 1985, 'Nigel: A Systemic Grammar for Text Generation'. In: R. Benson and J. Greaves (eds.): *Systemic Perspectives on Discourse: Selected Papers Papers from the Ninth International Systemics Workshop*. Ablex, London. Also available as USC/ISI Research Report RR-83-105.

Mann, W. C. and S. A. Thompson: 1987, 'Rhetorical Structure Theory: A Theory of Text Organization'. In: L. Polanyi (ed.): *The Structure of Discourse*. Ablex Publishing Corporation, Norwood, N.J. Also available as USC/Information Sciences Institute Technical Report Number RS-87-190.

Mastaglio, T. W.: 1990, *User Modelling in Cooperative Knowledge-Based Systems*. PhD thesis, Department of Computer Science, University of Colorado, Boulder.

Matthiessen, C. M. I. M.: 1984, 'Systemic Grammar in Computation: The Nigel Case'. In *Proceedings of the First Conference of the European Association for Computational Linguistics*, Pisa, Italy. European Association for Computational Linguistics. Also available as USC/ISI Research Report RR-84-121.

Mays, E.: 1980, 'Correcting Misconceptions about Data Base Structure'. In *Proceedings 3-CSCSI*, Victoria, B. C. Canadian Society of Computational Studies of Intelligence.

McCoy, K. F.: 1985, *Correcting Object-Related Misconceptions*. PhD thesis, University of Pennsylvania, December. Published by University of Pennsylvania as Technical Report MS-CIS-85-57.

McCoy, K. F.: 1988, 'Reasoning on a Highlighted User Model to Respond to Misconceptions'. *Computational Linguistics* **14**(3), 52–63.

McKeown, K. R.: 1985, *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press, Cambridge, England.

McKeown, K. R.: 1988, 'Generating Goal-Oriented Explanations'. *International Journal of Expert Systems* **1**(4), 377–395.

Mittal, V. and C. Paris: 1992, 'Generating Object Descriptions: Integrating Examples with Text'. In *Proceedings of the 1992 Canadian Artificial Intelligence Conference*. Canadian AI.

Moore, J. D. and C. L. Paris: 1989, 'Planning Text For Advisory Dialogues'. In *Proceedings of the Twenty-Seventh Annual Meeting of the Association for Computational Linguistics*, pp. 203–211, Vancouver, B.C., Canada, June 26–29.

Moore, J. D. and C. L. Paris: forthcoming, 'Planning Text for Advisory Dialogues: Capturing Intentional, Rhetorical and Attentional Information'.

Moore, J. D. and W. R. Swartout: 1989, 'A Reactive Approach to Explanation'. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 20–25, pp. 1504–1510.

Moore, J. D. and W. R. Swartout: 1990, 'Pointing: A Way Toward Explanation Dialogue'. In *Proceedings of the National Conference on Artificial Intelligence*, Boston, MA, July 29–August 3, pp. 457–464.

Moore, J. D.: 1989a, *A Reactive Approach to Explanation in Expert and Advice-Giving Systems*. PhD thesis, University of California, Los Angeles.

Moore, J. D.: 1989b, 'Responding to "Huh?": Answering Vaguely Articulated Follow-Up Questions'. In *Proceedings of the Conference on Human Factors in Computing Systems*, Austin, Texas, April 30–May 4, pp. 91–96.

Morik, K. and C.-R. Rollinger: 1985, 'The Real Estate Agent – Modeling Users by Uncertain Reasoning'. *AI Maganize* **6**, 44–52.

Neches, R., W. R. Swartout, and J. D. Moore: 1985, 'Enhanced Maintenance and Explanation of Expert Systems Through Explicit Models of Their Development'. *IEEE Transactions on Software Engineering* **SE-11**(11), 1337–1351.

Paris, C. L.: 1987, *The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise*. PhD thesis, Columbia University. To be published in the

"Communication in Artificial Intelligence" series, Steiner and Fawcett (eds.): Frances Pinter, 1992.

Paris, C. L.: 1988, 'Tailoring Object Descriptions to the User's Level of Expertise'. *Computational Linguistics* **14**(3), 64–78.

Paris, C. L.: 1991, 'Generation and Explanation: Building an Explanation Facility for the Explainable Expert Systems Framework'. In: C. L. Paris, W. R. Swartout, and W. C. Mann (eds.): *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Boston, Dordrecht, London: Kluwer Academic Publishers, pp. 49–81.

Pollack, M. E.: 1986a, 'A Model of Plan Inference that Distinguishes between the Beliefs of Actors and Observers'. In *Proceedings of the ACL-86*, pp. 207–214. Association of Computational Linguistics.

Pollack, M. E.: 1986b, *Inferring Domain Plans in Question-Answering*. PhD thesis, University of Pennsylvania. Published by University of Pennsylvania as Technical Report MS-CIS-86-40.

Quilici, A., D. Michael, and F. Margot: 1988, 'Providing Explanatory Responses to Plan-Oriented Misconceptions'. *Computational Linguistics* **14**(3), 38–51.

Quilici, A.: 1990, 'The Correction Machine: Using Common-Sense Planning Knowledge to Construct and Exploit User Models'. In *Proceedings of the Second International Workshop on User Modeling*. AAAI and the University of Hawaii.

Reithinger, N.: 1987, 'Generating Referring Expressions and Pointing Gestures'. In: G. Kempen (ed.): *Natural Language Generation: Recent Advances in Artificial Intelligence, Psychology, and Linguistics*. Boston, Dordrecht: Kluwer Academic Publishers, pp. 71–81.

Rich, E.: 1989, 'Stereotypes and User Modelling'. In: A. Kobsa and W. Wahlster (eds.): *User Models in Dialog Systems*. Symbolic Computation Series. Berlin, Heidelberg, New York, Tokyo: Springer-Verlag.

Ringle, M. H. and B. C. Bruce: 1981, 'Conversation Failure'. In: W. G. Lehnert and M. H. Ringle (eds.): *Knowledge Representation and Natural Language Processing*. Hillsdale, New Jersey: Lawrence Erlbaum Associates, pp. 203–221.

Sacerdoti, E. D.: 1975, 'A Structure for Plans and Behavior'. Technical Report TN-109, SRI.

Schuster, E., D. Chin, R. Cohen, A. Kobsa, K. Morik, K. Sparck Jones, and W. Wahlster: 1988, 'Discussion Section on the Relationship Between User Models and Discourse Models'. *Computational Linguistics* **14**(3), 5–22.

Shifroni, E. and B. Shanon: 1992, 'Interactive User Modeling: An Integrative Explicit-Implicit Approach'. *User Modeling and User-Adapted Interaction* **2**, 331–365 (this issue).

Sleeman, D. H. and J. S. Brown (eds.): 1981, *Intelligent Tutoring Systems*. London: Academic Press.

Sleeman, D. H.: 1983, 'Inferring Student Models for Intelligent Computer-Aided Instruction'. In: R. S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.): *Machine Learning: An Artificial Intelligence Appproach*. Tioga.

Sleeman, D. H.: 1985, 'UMFE: A User Modelling Front End SubSystem'. *International Journal of Man-Machine Studies* **23**, 71–88.

Sparck Jones, K.: 1984, 'User Models and Expert Systems'. Technical Report No. 61, University of Cambridge Computer Laboratory.

Sparck Jones, K.: 1989, 'Realism about User Modelling'. In: A. Kobsa and W. Wahlster (eds.): *User Models in Dialog Systems*. Symbolic Computation Series. Berlin, Heidelberg, New York, Tokyo: Springer-Verlag.

Stevens, A., A. Collins, and S. E. Golding: 1979, 'Misconceptions in Student's Understanding'. *International Journal of Man-Machine Studies* **11**, 145–156.

Swartout, W. R. and S. W. Smoliar: 1987, 'On Making Expert Systems More Like Experts'. *Expert Systems* **4**(3).

Swartout, W. R., C. L. Paris, and J. D. Moore: 1991, 'Design for Explainable Expert Systems'. *IEEE Expert* **6**(3), 58–64.

van Beek, P.: 1986, 'A Model for User Specific Explanations from Expert Systems'. Technical Report CS-86-42, University of Waterloo.

Wenger, E.: 1987, *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. Los Altos, CA: Morgan Kaufmann Publishers.

Wolz, U., K. R. McKeown, and G. E. Kaiser: 1990, 'Automated Tutoring in Interactive Environments: A Task-Centered Approach'. *Machine-Mediated Learning* **3**(1), 53–79.

Wu, D.: 1991, 'Active Acquisition of User Models: Implications for Decision-Theoretic Dialog Planning and Plan Recognition'. *User Modeling and User-Adapted Interaction* **1**(2), 149–172.