# A Reactive Approach to Explanation

Johanna D. Moore
University of California, Los Angeles
and
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

William R. Swartout
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

## Abstract

Explanation is an interactive process, requiring a dialogue between advice-giver and advice-seeker. Yet current expert systems cannot participate in a dialogue with users. In particular these systems cannot clarify misunderstood explanations, elaborate on previous explanations, or respond to follow-up questions in the context of the on-going dialogue. In this paper, we describe a reactive approach to explanation - one that can participate in an on-going dialogue and employs feedback from the user to guide subsequent explanations. Our system plans explanations from a rich set of explanation strategies, recording the system's discourse goals, the plans used to achieve them, and any assumptions made while planning a response. This record provides the dialogue context the system needs to respond appropriately to the user's feedback. We illustrate our approach with examples of disambiguating a follow-up question and producing a clarifying elaboration in response to a misunderstood explanation.

## 1 Introduction

Explanation requires a dialogue. Users need to be able to ask follow-up questions if they do not understand an explanation or want further elaboration. Answers to such questions must take into account the dialogue context. Studies of advisory consultations between humans bear out this observation, showing that explanation is an interactive process between explainer and advice-seeker [Pollack *et al.*, 1982]. Studying student-teacher interactions, we found that advice-seekers frequently did not fully understand the instructor's response. They frequently asked follow-up questions requesting clarification, elaboration, or re-explanation. In some cases, follow-up questions took the form of a well-articulated query; in other cases, the follow-up was a vaguely articulated mumble or sentence fragment. Often the instructor did not have much to go on, but still had to provide an appropriate response.

The research described in this paper was supported by the Defense Advanced Research Projects Agency (DARPA) under a NASA Ames cooperative agreement number NCC 2-520.

Unfortunately, current expert systems cannot participate in a dialogue with users. In particular these systems cannot clarify misunderstood explanations, elaborate on previous explanations, or respond to follow-up questions in the context of the on-going dialogue. In part, the explanation components of current expert systems are limited because they are quite simple. However, even the more sophisticated generation techniques employed in computational linguistics are inadequate for responding to follow-up questions. The problem is that *both* expert system explanation and natural language generation systems view generating responses as a *one-shot process.* That is, a system is assumed to have one opportunity to produce a response that the user will find satisfactory.

This one-shot approach is clearly inconsistent with analyses of naturally occurring advisory dialogues. Moreover, if a system has only one opportunity to produce a text that achieves the speaker's goals without over- or under-informing, boring or confusing the listener then that system must have an enormous amount of detailed knowledge about the listener. This has led to a view that improvements in explanation will come from improvements in the user model and considerable effort has been expended in representing a *detailed* model of the user - including the user's goals, what the user knows about the domain, how information should be presented to that user, and so forth [Appelt, 1981, McCoy, 1985, Paris, 1988, Kass and Finin, 1989]. However, following Sparck Jones [Sparck Jones, 1984], we question whether it will be possible to build complete and correct user models. Further, by focusing on user models, researchers have ignored the rich source of guidance that people use in producing explanations, namely feedback from the listener [Ringle and Bruce, 1981]. By throwing out the one-shot assumption, we can make use of that guidance.

Thus, a *reactive* approach to explanation is required - one in which feedback from the user is an integral part of the explanation process. A reactive explanation facility should include the ability to: 1) accept feedback from the listener, 2) recover if the listener indicates he is not satisfied with the response, 3) answer follow-up questions taking into account previous explanations, *not* as independent questions, 4) offer further explanations even if the user does not ask a well-formulated follow-up question, and 5) use information in a user model if it exists, but not require it.

## 2 Limitations of Current Systems

There are three main reasons why current systems cannot participate in a dialogue with their users. First, to be able to clarify a misunderstood explanation or respond to a follow-up question in context, a speaker must understand the explanation he has produced. Unfortunately, current expert systems produce explanations by filling in templates or using canned text and thus have little or no "understanding" of their own explanations. They do not represent the goal of the explanation, what rhetorical purposes are served by individual clauses in the text, or what assumptions about the listeners knowledge may have been made.

Second, current systems always interpret questions in the same way. For example, when a user types "why?" in response to a question, MYCIN assumes that the user is asking what higher domain goal gave rise to the current one. Subsequent "whys" retrieve information about still higher level domain goals that posted the ones just described. As we describe later, this single interpretation fails to take into account the dialogue context and can be unnatural.

The third problem with current systems is that they typically have only a single response strategy associated with each question type. Making oneself understood often requires the ability to present the same information in multiple ways or to provide different information to illustrate the same point. Without multiple strategies for responding to a question, a system cannot offer an alternative response even if it understands why a previous explanation was not satisfactory.

We have built an explanation component for an expert system which addresses these problems. To provide the capabilities described above, we: 1) plan responses such that the intentional structure of the responses is explicit and can be reasoned about, 2) keep track of conversational context by remembering not only what the user asks, but also the planning process that led to an explanation, 3) taxonomize the types of (follow-up) questions that are asked and understand their relationship to the current context, and 4) provide flexible explanation strategies with many and varied plans for achieving a given discourse goal.

## 3 System Description

Our explanation generation facility is part of the Explainable Expert Systems (EES) framework [Neches *et al.*, 1985]. When an expert system is built in EES, an extensive development history is created that records the domain goal structure and design decisions behind the expert system. This structure is available for use by the explanation facility.

We have used EES to construct a prototype expert system, the Program Enhancement Advisor (PEA) [Neches *et al.*, 1985], which we are using as a testbed for our work on explanation generation. PEA is an advice-giving system intended to aid users in improving their Common Lisp programs by recommending transformations that enhance the user's code.[1] The user supplies PEA with the program to be enhanced. PEA begins the dialogue with the user by asking what characteristics *of* the program he would like to improve. The user may choose to enhance any combination of readability, maintainability, and efficiency. PEA then recommends transformations that would enhance the program along the chosen dimensions. After each recommendation is made, the user is free to ask questions about the recommendation.

An overview of the explanation generation facility (and its relation to the PEA expert system) is shown in Figure 1. The *text planner* is central to the explanation facility. The planner uses a top-down hierarchical expansion planning mechansism. When a discourse goal is posted, the text planner searches its library of explanation strategies looking for strategies that can achieve it. A strategy is selected and may in turn post subgoal[c] for the planner to refine. Planning continues in this fashion until the entire plan is refined into primitive operators, i.e., speech acts such as INFORM, RECOMMEND. As the system plans explanations, it keeps track of any assumptions it makes about what the user knows as well as alternative strategies that could have been used to achieve the discourse goals. The result is a *text plan* for achieving the original discourse goal. This text plan is recorded in the *dialogue history* and passed to the Penman text generation system [Mann and Matthiessen, 1983] for translation into English.

A discourse goal may be posted as a result of reasoning in the expert system or as a result of a query from the user. User queries must first be interpreted by the *query analyzer.* Even though we assume the user poses queries in a stylized notation,[2] ambiguities may still arise. An example of an ambiguous follow-up question and the process we use to disambiguate it appears in Section 4.1.

Input to the query analyzer may be a follow-up question (e.g. "Why?", "What is a generalized-variable?"), an indication that the user does not understand the system's response ("Huh?"), or an indication that the user understands and has no follow-up question ("Go Ahead"). The query analyzer interprets this feedback and either returns control to the expert system, or formulates the appropriate discourse goal and passes it to the text planner to produce a response.

If the user asks a follow-up question or indicates that he does not understand the explanation, the system examines the dialogue history. The information contained there concerning the goal structure of the explanation, assumptions made during its generation, and alternative strategies, is necessary in disambiguating follow-up questions, selecting perspective when describing or com-

[1]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.

[2]To avoid the myriad problems of parsing English-input, we require that the user's questions be posed in a stylized language. We have also provided a "mouse" interface that allows a user to point to parts of the system's explanations that he doesn't understand or has questions about.

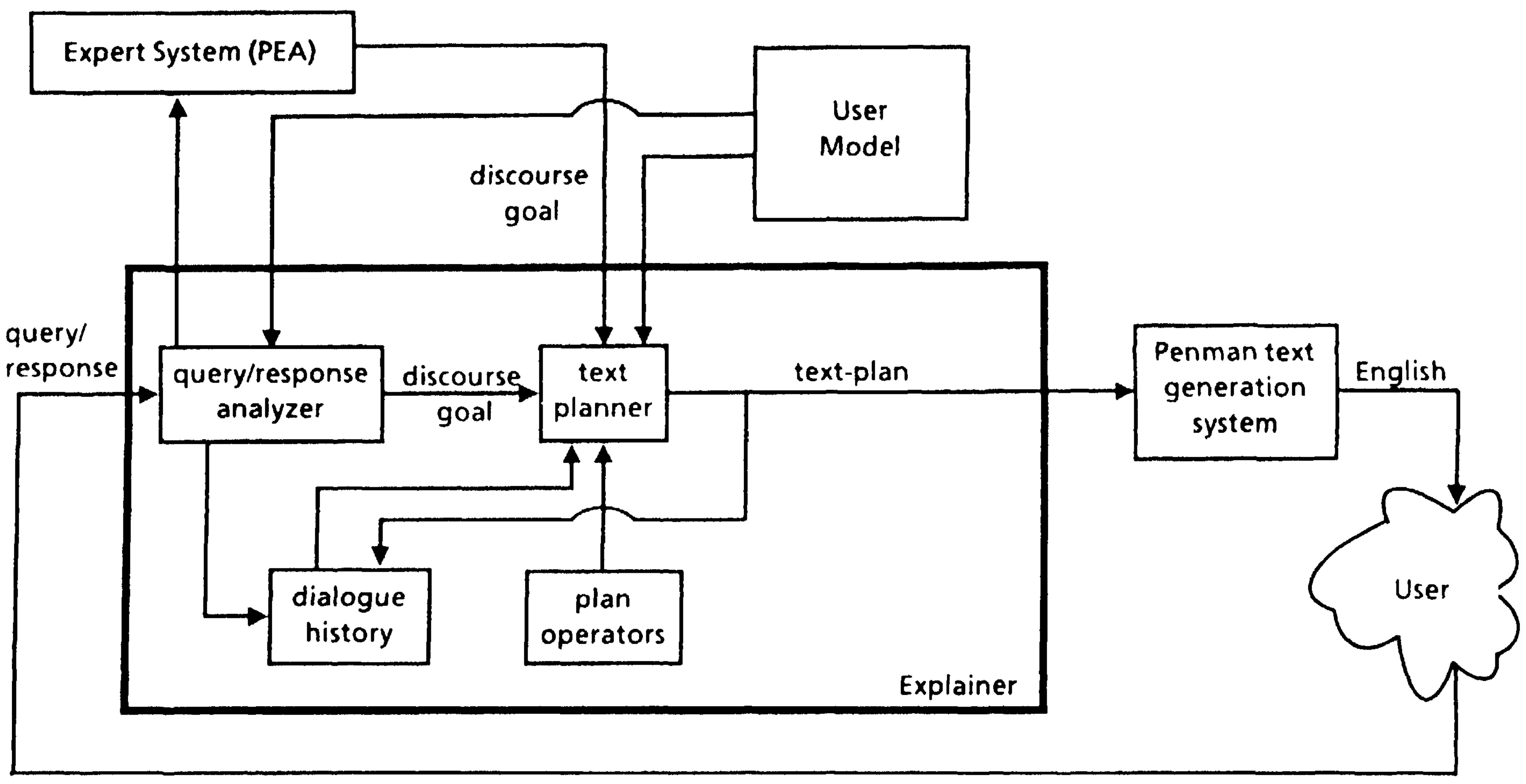

Figure 1: Architecture of Explanation System

paring objects, and clarifying misunderstandings.

## 4 Examples

Consider the sample dialogue with our system shown in Figure 2. While enhancing maintainability, the system recommends that the user perform an act, namely replace setq with setf.[3] The user, not immediately convinced that this replacement should be done, asks "why?". The query analyzer interprets this question and posts the goal: (PERSUADE S H (GOAL H Eventually(DONE H replace-1))) where S is the speaker, H is the hearer, and replace-1 is the act of replacing setq with setf. This is a goal to persuade the user to perform replace-1. Discourse goals are represented in terms of the effects that the speaker intends his utterance to have on the hearer.[4]

When a discourse goal is posted, the text planner searches for operators capable of satisfying it, i.e., all operators whose *Effect* matches the goal. Each plan operator also contains a *Constraint* list, which limits the applicability of the operator; a *Nucleus,* which is a discourse goal for the main topic to be expressed; and optionally, a list of *Satellites* which are discourse goals that express additional information needed to achieve the Effect of the operator.

One of the plan operators that matches the current goal is shown in Figure 3.[5] Informally, this plan operator states that if an act is a step in achieving some domain goal(s) that the hearer shares, then one way to persuade the hearer to do the act is to motivate the act in terms of those goals. More formally, this operator's constraints require that there be a *?domain-goal* such that: *?domain-goal* is a goal of the expert system, replace-1 is a step in achieving *? domain-goal,* and the speaker and hearer mutually believe that *?domain-goal* is a goal of the hearer. In order to bind *?domain-goal,* the text planner examines the expert system's goal structure. The system assumes that the user shares its top-level goal, enhance-program, since he is using the system to perform that task. Furthermore, since the system asks what characteristics the user would like to enhance, the system can assume that the user shares the goal of enhancing those characteristics; in this case, enhance-maintainability. The information that the user shares these two domain goals is included in the *user model.* In order to avoid explaining parts of the reasoning chain that the user is familiar with, the more specific goal is chosen. In this example, once the constraints have been satisfied, the only possible binding for the variable *?domain~goal* is enhance-maintainability.

Once a plan operator has been selected, the planner instantiates it by posting its Nucleus and required Satellites as subgoals to be refined. In this case, since there is only one binding for *?domain-goal,* the single subgoal (MOTIVATION replace-1 enhance-maintainability) is posted. One strategy for satisfying this goal, shown in Figure 4, is to inform the hearer of the goal the system

In many instances PEA is capable of performing the transformation. In such cases, while the actual replacement is done by the system, the user's approval is required.

[4]Following Hovy [Hovy, 1988], we use the terminology for expressing beliefs developed by Cohen and Levesque in their theory of rational interaction [Cohen and Levesque, 1985]. Space limitations prohibit an exposition of their terminology in this paper. We provide English paraphrases where necessary for clarity.

[5](BMB S H x) should be read as "S believes that S and
H mutually believe x." Our plan language makes use of Rhetorical Structure Theory [Mann and Thompson, 1988], a descriptive theory characterizing text structure in terms of the relations (e.g. MEANS, MOTIVATION) that hold between parts of a text. A detailed description of the plan language is beyond the scope of this paper, see [Moore and Paris, 1989].

| | | |
|---|---|---|
| SYSTEM | What characteristics of the program would you like to enhance? | [1] |
| USER | Maintainability. | [2] |
| | ⋮ | |
| SYSTEM | You should replace `(setq x 1)` with `(setf x 1)`. | [3] |
| USER | Why? | [4] |
| SYSTEM | I'm trying to enhance the maintainability of the program by applying transformations that enhance maintainability. **Setq-to-setf** is a transformation that enhances maintainability. | [5] |
| USER | Why? | [6] |
| SYSTEM | Transformations that enhance maintainability are defined as transformations 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. | [7] |
| USER | What is a generalized-variable? | [8] |
| SYSTEM | Simple-variables are storage locations named by symbols. This concept can be generalized to a storage location that can be named by any accessor function. These storage locations are called generalized-variables. | [9] |
| USER | Huh? | [10] |
| SYSTEM | For example, the *car* and *cdr* of a cons are generalized-variables, named by the accessor functions **car** and **cdr**. Other examples are an element of an array or a component of a structure. | [11] |

Figure 2: Sample Dialogue

```
EFFECT: (PERSUADE S H (GOAL H Ev(DONE H ?act)))
CONSTRAINTS: (AND (GOAL S ?domain-goal)
                  (STEP ?act ?domain-goal)
                  (BMB S H (GOAL H ?domain-goal)))
NUCLEUS: (FORALL ?domain-goal (MOTIVATION ?act ?domain-goal))
```

Figure 3: Plan Operator for Persuading User to Do an Act

```
EFFECT: (MOTIVATION ?act ?domain-goal)
CONSTRAINTS: (AND (GOAL S ?domain-goal)
                  (STEP ?act ?domain-goal))
NUCLEUS: (INFORM S H ?domain-goal)
SATELLITES: (((MEANS ?domain-goal ?act)))
```

Figure 4: A Plan Operator for Motivating an Act

is trying to achieve (the Nucleus) and then to establish that the act in question is part of the means for achieving the goal (the Satellite). These subgoals are eventually refined to speech acts. The final text plan, shown in Figure 5, is added to the dialogue history and passed to the generator which produces response (5) in the sample dialogue.

## 5 Disambiguating Follow-up Questions

After this response is presented, the user asks "why?" a second time. At this point, there are several possible interpretations of this question, including:

I1: Why are you trying to enhance the maintainability of the program?

I2: Why are you trying to enhance the maintainability of the program by applying transformations that enhance maintainability? (as opposed to enhancing the program via some other method)

I3: Why are you applying transformations that enhance maintainability?

I4: Why is setq-to-setf a transformation that enhances maintainability?

Recall that in cases such as this, MYCIN always assumes that "why" is asking why the system is trying to achieve the higher-level domain goal, corresponding to interpretation I1. This interpretation is often inappropriate. Users are frequently asking for justification of factual statements made in the explanation, corresponding to I4. Even if MYCIN could recognize the multiple interpretations, it could not decide among them because it does not maintain a dialogue history and does not understand the responses it generates.

Resolving ambiguity requires: 1) identifying candidate interpretations, and 2) choosing among them. While these tasks are conceptually distinct, our system interleaves them to increase efficiency. It generates the most likely interpretations first, and then uses heuristics to rule them out. If an interpretation cannot be ruled out, it is chosen as the interpretation of the user's question and a response is generated. If the interpretation is incorrect, the user can still recover by asking a follow-up

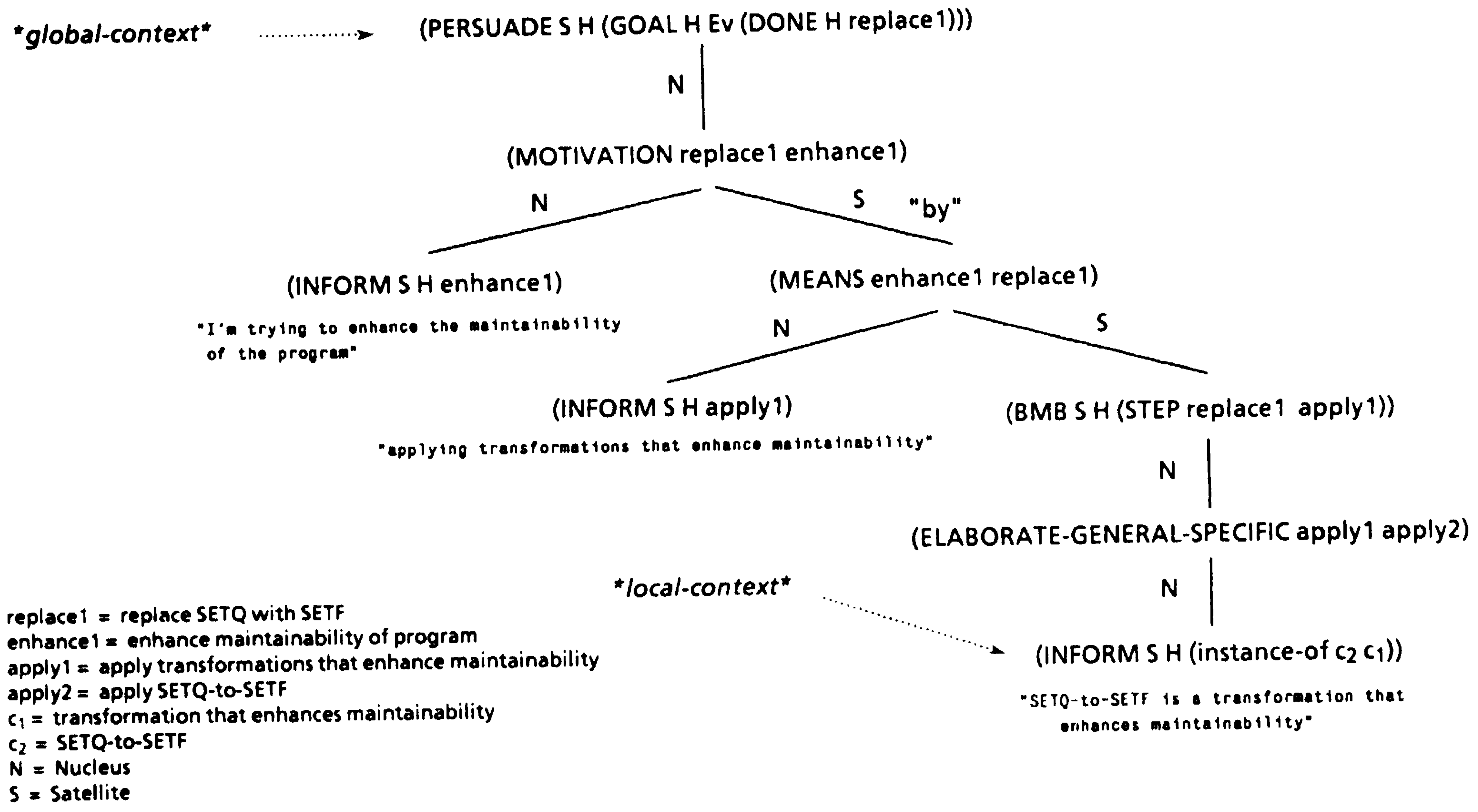
*global-context*
(PERSUADE S H (GOAL H Ev (DONE H replace1)))
N
(MOTIVATION replace1 enhance1)
N
S "by"
(INFORM S H enhance1)
"I'm trying to enhance the maintainability of the program"
(MEANS enhance1 replace1)
N
S
(INFORM S H apply1)
"applying transformations that enhance maintainability"
(BMB S H (STEP replace1 apply1))
N
(ELABORATE-GENERAL-SPECIFIC apply1 apply2)
*local-context*
N
(INFORM S H (instance-of c2 c1))
"SETQ-to-SETF is a transformation that enhances maintainability"
replace1 = replace SETQ with SETF
enhance1 = enhance maintainability of program
apply1 = apply transformations that enhance maintainability
apply2 = apply SETQ-to-SETF
c1 = transformation that enhances maintainability
c2 = SETQ-to-SETF
N = Nucleus
S = Satellite

Figure 5: Completed Text Plan for Persuading the User to Replace SETQ with SETF

question.

Our system uses the following heuristics to identify and choose among multiple interpretations:

Hi: Follow immediate focus rules (continuing on the same topic is preferred over returning to a previously mentioned topic, [Sidner, 1979, McKeown, 1982].)

H2: Don't tell the user things he already knows.

H3: Don't tell the user things you've already said.

We have found that focus of attention (HI) is a powerful heuristic for ordering the generation of likely interpretations. In the current example, the most recent focus of attention (indicated by the * *local-context*)* is the statement Setq-to-setf is a transformation that enhances maintainability. The system thus infers that the question concerns the rationale behind this statement (14), unless H2 or H3 rules that interpretation out. For example, that interpretation could be ruled out by 112 if the user model indicated that the user knew why setq-to-setf enhances maintainability. In our example, nothing rules out 14, so the system explains why setq-to-setf enhances maintainability.

If the first interpretation is ruled out, the system uses the next most recent focus of attention[6] to form the next possible interpretation. In this example, that focus refers to the method the system is applying to achieve a goal (i.e., applying transformations that enhance maintainability). This leads to interpretation 13. This interpretation will be ruled out by H3 because from the semantics of the rhetorical relation MEANS, we determine that we have just told the user that the system is using the method of applying transformations that enhance maintainability in order to achieve the goal enhance maintainability.

The next most recent focus refers to the means by which the enhancement is being achieved (i.e., by applying transformations that enhance maintainability). This leads to interpretation 12. If that interpretation were also ruled out, the system would continue in this fashion until it found an acceptable interpretation or reached the global context, which, in this case, is the top node in the text plan.

It is interesting to note that the focus of the user's question is derived from the *system's* statement, *not* the user's. The user simply types "why?" in his first two queries in the dialogue; context comes from the response generated by the system. Until now, much work has concentrated on building discourse models that keep track of the user's goals and plans - both domain goals [Carberry, 1983, McKeown *et al,* 1985] and discourse goals [Litman, 1985]. Little work has been done on keeping track of the system's discourse goals and the plans it uses to achieve them.[7] As our example illustrates, conversational context must include the system's statements.

[6]The text plan records the order in which topics appear in the explanation. This information is used to derive foci of attention in order. In choosing interpretations of "Why?", the system skips over certain rhetorical relations in the text plan that are considered to be purely presentational in nature, e.g., ELABORATE, BACKGROUND.

[7]Recent work by Grosz and Sidner addresses this issue for the purposes of analyzing dialogues [Grosz and Sidner, 1986].

```
EFFECT: (BMB S H (KNOW H ?concept))
CONSTRAINTS: (AND (SUBCLASS ?sub-concept ?concept)
                  (BMB S H (KNOW H ?sub-concept))
                  (IMMEDIATE-SUBCLASS ?concept ?super-concept))
NUCLEUS: ((SETQ ?diffs (ESSENTIAL-DIFFERENCES ?sub-concept ?concept))
                  (BMB S H (DETAILS-OF ?subconcept ?super-concept ?diffs)))
SATELLITES: (((ABSTRACTION ?sub-concept ?concept ?super-concept ?diffs)))
```

Figure 6: Plan Operator for Describing an Object by Abstraction

## 6 Answering a Vaguely Articulated Follow-Up Question

The user then asks the question, "What is a generalized-variable?". The query analyzer interprets the question and posts the discourse goal (BMB S H (KNOW H generalized-variable)), i.e., the speaker wishes to achieve the state where the speaker and hearer mutually believe that the hearer knows the concept generalized-variable.

The system has several plan operators for achieving such a goal. It may describe a concept by describing its attributes and its parts, by drawing an analogy with a similar concept, by giving examples of the concept, or by generalizing a concept the user is familiar with. The plan operator for the latter is shown in Figure 6.

To choose from among these candidate plan operators, the planner has several selection heuristics, including:

SHI: Prefer operators that require making no assumptions about the hearer's beliefs.

SH2: Prefer operators that make use of a concept the hearer knows.

SH3: Prefer operators that make use of a concept mentioned in the dialogue history.

In this case, the user model indicates that the hearer knows the concept simple-variable. Hence the operator in Figure 6 requires making no assumptions about the hearer's knowledge, makes use of a concept the user knows, and uses a concept previously mentioned in the dialogue. Thus it is ranked highest by the plan selection heuristics. The final text plan for this example first describes simple-variables and then abstracts this concept to introduce generalized-variables. This produces the response shown in the sample dialogue.

The user then indicates that he does not understand this explanation with the vaguely-articulated follow-up, "Huh?". From our analysis of naturally occurring dialogues, we devised a set of *recovery heuristics* for responding to such a question. These include:

RHI: If the discourse goal is to describe a concept, give example(s).

RII2: If the discourse goal is to describe a concept, and there is an analogous entity that the hearer knows, draw an analogy to the familiar concept.

RH3: Expand any unexpanded optional satellites in previous plan operators.[8]

RH4: If another plan exists for achieving the discourse goal, try it.

[8]Plan operators may contain optional satellites which the system may decide to leave unexpanded during planning.

RH1 and RH2 apply in the context of a particular discourse goal, namely describing a concept, while the other heuristics are more general. The system tries to apply its most specific knowledge first. In this case, RH1 applies and the explainer recovers by giving examples.

As illustrated in Figure 6, constraints on plan operators often refer to the state of the hearer's knowledge. The user model includes the domain concepts and problem-solving knowledge, i.e., domain goals and plans, assumed to be known to the current user. However, the system does not require that this model is be either *complete* or *correct.* Therefore, the user model may contain concepts the user does not actually know or omit concepts the user does know. To satisfy a constraint on an operator, the system may assume that a concept is known to the user even if it is not indicated in the user model. As described above, when such an assumption is made, the selection heuristics give the operator a lower rating. If the operator is selected, the fact that an assumption was made is recorded in the plan structure. The system must keep track of such assumptions because these are likely candidates if a misunderstanding occurs later. This leads to another recovery heuristic:

RH5: If any assumptions were made in planning the last explanation, plan responses to make these assumptions true.

For example, in producing response (7) in Figure 2 the system assumed (erroneously) that the user knew what generalized-variables were and simply used it without defining it. In (8), the user asked explicitly for a definition. If he had just typed "Huh?" instead, the system would have examined its assumptions and used RH5 to plan a response defining generalized variables.

## 7 Current Status and Conclusions

The expert system and explanation facility described are implemented. There are approximately 75 plan operators, 5 plan selection heuristics, and 5 recovery heuristics. The system can produce the text plans necessary to participate in the dialogue shown and several others that are similar.

In summary, current expert systems fail to support explanation as a dialogue. Their unnatural, one-shot approach to explanation depends critically on the quality of the user model and is seriously degraded if that model is incomplete or incorrect. Because they fail to support dialogue, these systems cannot clarify misunderstood explanations, elaborate on previous explanations, or respond to follow-up questions in the context of the on-going dialogue.

As an alternative, we proposed a reactive model of explanation - in which the system can employ feedback from the user and participate in a dialogue. 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 listener indicates that an explanation was not understood. Our system can employ information in a user model when it is available, but is not critically dependent on that information.

## Acknowledgements

The authors would like to thank Cecile Paris for collaboration on the plan language and Margot Flowers, Neil Goldman and Jasmina Pavlin for comments on earlier versions of this paper.

## References

[Appelt, 1981] Douglas E. Appelt. *Planning Natural Language Utterances to Satisfy Multiple Goals.* PhD thesis, Stanford University, 1981.

[Carberry, 1983] Sandra Carberry. Tracking user goals in an information-seeking environment. In *Proceedings of the Third National Conference on Artificial Intelligence,* pages 59-63, Washington, D.C., August 22-26 1983.

[Cohen an d Levesque, 1985] Philip R. Coh en and Hector J. Levesque. Speech acts and rationality. In *Proceedings of the Twenty-Third Annual Meeting of the Association for Computational Linguistics,* pages 49-60, University of Chicago, Chicago, Illinois, July 8-12 1985.

[Grosz and Sidner, 1986] Barbara J. Grosz and Candace L. Sidner. Attention, intention, and the structure of discourse. *Computational Linguistics,* 12(3): 175—204,1986.

[Hovy, 1988] Eduard H. Hovy. Planning coherent multisentential text. In *Proceedings of the Twenty-Sixth Annual Meeting of the Association for Computational Linguistics,* State University of New York, Buffalo, New York, 7-10 June 1988.

[Kass and Finin, 1989] Robert Kass and Tim Finin. Modeling the user in natural language systems. *Computational Linguistics Journal,* 14:5-22, 1989. Special Issue on User Modelling.

[Litman, 1985] Diane Litman. *Plan Recognition and Discourse Analysis: An Integrated Approach for Understanding Dialogues.* PhD thesis, University of Rochester, 1985. Published by University of Rochester as Technical Report TR 170.

[Mann and Matthiessen, 1983] William C. Mann and Christian Matthiessen. Nigel: A systemic grammar for text generation. Technical Report, RR-83-105, USC/Information Sciences Institute, February 1983.

[Mann and Thompson, 1988] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: Towards a functional theory of text organization. *TEXT,* 8(3):243-281, 1988.

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

[McK eown *et al.,* 1985] Kathleen R. McKeown, Myron Wish, and Kevin Matthews. Tailoring explanations for the user. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* pages 794-798, Los Angeles, CA, August 1985. IJCAI.

[McKeown, 1982] Kathleen R. McKeown. *Generating Natural Language Text in Response to Questions About Database Structure.* PhD thesis, University of Pennsylvania, 1982. Published by University of Pennsylvania as Technical Report MS-CIS-82-5.

[M oore and Paris, 1989] Johanna D. Moore and Cecile L. Paris. Planning text for advisory dialogues. In *Proceedings of the Twenty-Seventh Annual Meeting of the Association for Computational Linguistics,* Vancouver, B.C., Canada, June 26-29 1989.

[Neches *et al,* 1985] Robert Neches, William R. Swartout, and Johanna D. Moore. Enhanced maintenance and explanation of expert systems through explicit models of their development. *IEEE Transactions on Software Engineering,* SE-11(11), November 1985.

[Paris, 1988] Cecile L. Paris. Tailoring object descriptions to a user's level of expertise. *Computational Linguistics Journal,* 14(3), 1988.

[Pollack *et al.,* 1982] Martha E, Pollack, Julia Hirschberg, and Bonnie Lynn Webber. User participation in the reasoning processes of expert systems. In *Proceedings of the Second National Conference on Artificial Intelligence,* Pittsburgh, Pennsylvania, August 18-20 1982.

[Ringle and Bruce, 1981]
Martin H. Ringle and Bertram C. Bruce. Conversation failure. In Wendy G. Lehnert and Martin H. Ringle, editors, *Knowledge Representation and Natural Language Processing,* pages 203-221. Lawrence Earlbaum Associates, Hillsdale, New Jersey, 1981.

[Sidner, 1979] Candace L. Sidner. *Toward a Computational Theory of Definite Anaphora Comprehension in English Discourse.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Mass., 1979.

[Sparck Jones, 1984] Karen Sparck Jones. User models and expert systems. Technical Report No. 61, University of Cambridge Computer Laboratory, December 1984.