Finding a Better Way: Choosing and Explaining Alternative Plans

Ursula Wolz*

Technical Report CUCS-409-88

Columbia University
Department of Computer Science
New York, NY 10027

Abstract

Many users of interactive computing environments have been confronted with the dilemma: "I know there must be a better way to do what I'm doing, but I don't know what it is." This paper describes a solution to this problem and several related ones through an automated consultant called GENIE, a system that answers users' questions about how to accomplish tasks. Our approach is based on the principle that the best plan to tell a user is not always computationally optimal, but is the plan most suitable to what the user already knows in the current context. To find a best plan, GENIE explores a declarative representation of goals that explicitly encodes alternative plans for goals, and the semantic relationships between alternatives. Both an Expert and a User model are represented in this manner, which allows us to abandon stereotypes of user expertise and functional difficulty of the domain constructs. The criteria for each choice of sub-step of a plan is based on two sets of heuristics applied to the semantic relationships encoded between plans. The first set is dependent upon what the user is currently doing, the second on what the user has done in the past.

Copyright © 1989 Ursula Wolz

*This Research is supported in part by ONR grant N00014-82-K-0256, by NSF grant IST-84-51438, a grant from DARPA, and a grant from Siemens Research and Technology Laboratories.

1. Introduction

Many users of interactive computing environments have been confronted with the dilemma: "I know there must be a better way to do what I'm doing, but I don't know what it is." This paper describes how we address this problem and several related ones through an automated consultant called GENIE (GENerated Informative Explanations), a system that answers users' questions about how to accomplish tasks in the domain of Berkeley Unix™ Mail. Our approach is based on the principle that the best plan to tell a user for accomplishing a task is not necessarily the optimal one. Rather, it is the plan most suitable in the current context and based on what the user already knows.

For example, there are at least two ways in most mail systems to send a message to a set of people. One can type each address in turn when prompted for the receiver of the message. One can also create an *alias* which is a named list of addresses that can be reused, and type the alias name at the prompt. The first method is most appropriate when the set occurs only in this instance, or is very small and easy to remember. The second method is more appropriate if over a period of time many messages will be sent to this set of people. Furthermore, a user who is new to sending messages may be overwhelmed by hearing about aliases, even if the message is to be sent to a group of users.

This paper presents a component of GENIE called the Plan Analyst (PA) that can find "better plans" for users' tasks (their computation goals.) Given our initial dilemma, the PA must not only be able to generate a plan, it must be able to compare plans with specific criteria for choosing one over the other. We have found that the method for choosing a "better way", is also applicable to choosing "any" way, that is when a user just asks how to do something. It is also useful when the user's plan doesn't work, both in diagnosing the problem and suggesting alternatives. Furthermore, there are times when the user may ask other kinds of questions, and GENIE is able to take the opportunity to suggest a better way. Therefore, regardless of the type of question asked, the PA must be able to produce a "best" plan for the situation.

The next section presents an overview of GENIE. Section 3 describes the PA in detail, while section 4 presents an example of how different plans are chosen depending on context and the user model. Section 5 is a discussion and summary.

2. Consulting In Interactive Environments

Interactive computing environments are inherently *procedural*, that is, they are used to *do tasks*. Therefore a consultant must be able to answer questions about how computational goals can be satisfied through plans, and how those plans are composed of sub-goals or of functions defined for the environment. We characterize question answering as a three stage process of understanding the user's question, analyzing what to say, and choosing how to say it. In procedural environments, choosing what to say requires analyzing plans.

To put the PA in perspective, Figure 2-1 shows the requisite components of GENIE which is being developed in Common Lisp on a Sun 3/60. The knowledge representations appear in boxes, the processes in boxes with rounded edges. We are primarily concerned with components of content generation, which are circled in grey. For demonstration and testing purposes we are also building simple understanding mechanisms and are using a surface text generator. The range of question intentions we are interested in reduces to those in Figure 2-2. Work by McKeown [7] and Paris [9] has addressed questions about domain objects, such as "What is a", and "What does it do." A question also identifies a goal or plan (or function) or both, and implies an assumption about their validity and the expected answer.

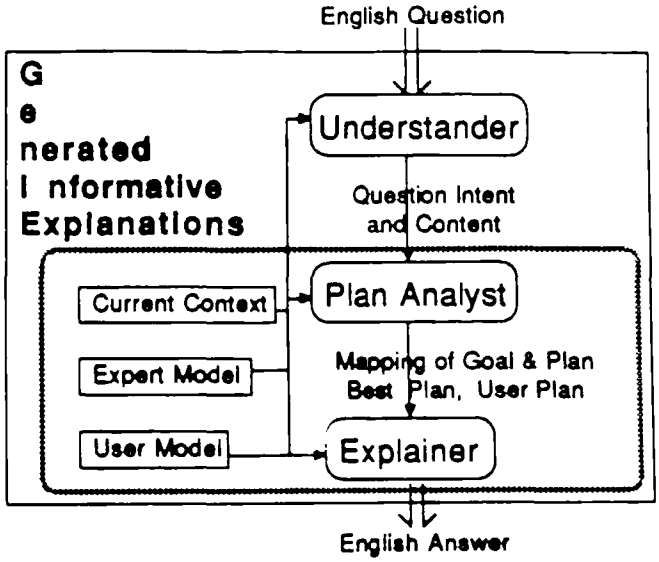

Figure 2-1: Components of the Prototype GECIE

The Understander derives the question intent from the form of the question. This is often termed the *discourse goal*, since it is the goal to elicit some information. It is distinguished from the computational goal of actually accomplishing some task in the environment. In what follows, the computational goal about which help is sought will be referred to as the *goal*, and the discourse goal as the *question intent*.

The Explainer chooses among four strategies to present the information to the user. It chooses to: **Introduce** — presenting functions and goals that the user has not encountered before, **Remind** — briefly describing functions and plans that the user has been exposed to but may have forgotten, **Clarify** — explaining distinctions and options about functions and plans, and **Elucidate** — clearing up misunderstandings that have developed about functions and plans. For example,

Question Intent	Identifies	Assumes	Expects
What plan satisfies this goal?	Goal	Goal is possible	Plan
What goal does this plan satisfy?	Plan	Plan satisfies some Goal	Goal
Does this plan satisfy this goal?	Plan & Goal	Plan satisfies goal	Match
This plan fails for this goal, what's wrong?	Plan & Goal	Plan should not satisfy Goal	Mismatch
Is there a better way to satisfy this goal?	Plan & Goal	Plan may not be best	Better P

Figure 2-2: Information Embedded in the Question Intention

if the user has never attempted to send mail to a group, GENIE may choose to *introduce* the plan. However, if GENIE knows the user has some inefficient plan, it should *clarify* the distinctions between the user's plan and a more efficient one. The strategies are based on an analysis of support materials and are discussed in [14]. They are similar to McKeown's text generation schemata [7].

The Explainer needs five parameters to form an answer. Some are derived by the Understander from the question, the others are determined by the PA:

- **A question intent - QI**, that provides an expected discourse focus.
- **A computational goal - G**, which is either identified in the question, or inferred by the PA.
- **A User Model plan - U**, which is the plan that the PA thinks the user has used in the past for accomplishing the goal.
- **A stated plan - S**, which only exists if stated in the question. It may not exist in the User Model — the user may have just learned it from somewhere else.
- **A best plan - B**, which is inferred by the PA from the Expert Model, given the current context and the the User Model. It may or may not be the same as S or U.

The domain knowledge of both a consultant and a user can be characterized as a *web* of interrelated goals for doing tasks, plans for accomplishing those goals, steps within plans that are either goals themselves (sub-goals), or functions that describe the actions available in the environment. The web is the basis of both the Expert Model and User Model. Presumably the former is considerably richer than the latter. The User Model's goals may contain plans that were attempted, but didn't work, or plans that do not exist in the Expert Model. In GENIE both models are frame-based structures that are searched in order to locate information. Computational goals contain a set of alternative plans for satisfying the goal, and semantic relational links that describe the distinctions between plans. A plan contains a sub-goal or an ordered sequence of sub-goals that specify how it can be executed.

Functions are degenerate plans that satisfies goals directly. They will often be implicitly included in later discussions of plans. Functions describe the operators of the environment. They encode the correct syntax of the function, any preconditions and effects, and the actions associated with parameters. Preconditions define a state that must be true before a function can be correctly executed. They may also contain a link to a goal that could satisfy it. Effects encode the actions of functions when applied to the Current Context.

The Current Context is a state space that includes the existence of and relationships between specific objects during an interactive session. It is represented as an add/delete list. In an electronic mail environment the objects include messages, users and collections of each.

From this perspective, expertise and difficulty of functions are characterized by the richness of the web, rather than as simple spectra as described by Chin [3]. In the model presented here, functions do not need to be classified as ''hard'' or ''easy'' since the difficulty of a function is dependent upon its role within a plan, not on a global taxonomy. Similarly, classifications such as ''novice'', ''intermediate'' and ''expert'' are unnecessary because users are judged by what they know about the current task, rather than how much they know about the entire environment. It is perfectly plausible for a user to have a rich web of knowledge about a portion of the environment, and almost no expertise about others. For example, a user may have extensive experience sending simple messages, and almost none with modifying messages through an editor.

Our frame-based perspective is distinguished from previous research in planning speech utterances [1, 2, 10, 11], that rely on a theorem prover and represent plans as predicates. The web, and the resulting search strategies of the PA form a middle ground between the complex reasoning that occurs during theorem proving and the simple selection from a *knowledge pool* used by McKeown [7] and Paris [9]. The web developed as a result of our need to both reason about plans and provide complex tutorial-like utterances.

3. The Plan Analyst

The Plan Analyst uses one of three search strategies in order to complete the parameter list for the Explainer. They are:

- Given G, but not S, attempt to find B and U.
- Given S, but not G, attempt to find G, B and U.
- Given S and G, find a relationship between them, and find B and U.

Given a goal, the Plan Analyst searches the Expert Model for the ''best'' plan for the goal using two sets of heuristic rules, one based on *world knowledge* and the other on the User Model. The choice between alternatives plans for each subgoal is made using a simple metric that scores all candidate plans found through world knowledge equally, and weighs user knowledge.

First the PA tries to choose steps in plans using world knowledge such as efficiency and temporality based on the Current Context. For example, users normally prefer plans that satisfy goals now rather than later, or that require fewer rather than more steps. In a mail environment the choice may be affected by whether a message, address or alias already exists. Whenever the PA encounters alternative plans for a goal, it uses the relational links between plans to reduce the set of rules to apply to the Current Context. For example, if the distinction between two plans has to do with the existence or lack of an object in the Current Context, then the PA shouldn't waste time exploring world knowledge pertaining to temporality. As the PA constructs the best plan it retains the relationships that mitigated its choices, so that the Explainer can include them in the answer.

The second set of heuristics compares the candidate plans to knowledge in the User Model. A weighting scheme is used in which the existence of the entire plan in the User Model scores best, and each level of depth searched decreases the worth of found components. Search in the User Model is aborted when the weight of any found components will only produce a lower score than those already achieved by a competing plan through world knowledge.

The Plan Analyst also returns other information, such as whether the best plan, the stated plan and the User Model plan are the same. If there is no user plan, or if it does not match the best plan, then the Plan Analyst also attempts to locate components of the stated plan and the best plan in the User Model.

Given a plan and a goal, the Plan Analyst uses the Expert Model to find a "match" between them. A match is defined as confirmation that a plan or function satisfies a goal. If it is unsuccessful it tries to find one of the following cases:

- A step in the plan has missing preconditions.
- A step in the plan is missing.
- A step in the plan is extraneous.
- A step in the plan that has missing preconditions is related to a step that is missing.
- A step that is missing is related to a step that is extraneous.
- A missing precondition is related to an extraneous step.

These are based on work by Joshi, Webber, and Weischedel [4], and a more detailed discussion of their use can be found in [12].

Given a plan, the Plan Analyst attempts to locate the goal it satisfies. If no actual match emerges from a list of candidates, then the goal with the least complex mismatch is chosen, but is marked as a mismatch along with its causes.

4. Examples of Choosing Between Plans

Two examples are presented that show how GENIE chooses the best plan. Others can be found in [13]. The first example illustrates how GENIE uses world knowledge to choose a plan. The second shows two responses to a user's request for a better plan, where the difference is based on the context and what the user knows. Figure 4-1 is a graphic representation of the Expert Model required to answer both questions.

First consider the question "How can I answer a message?" The question identifies a goal (*reply.to.message*) and has a question intent (QI) to receive a plan in response. Assume that the Current Context contains a message that the user is currently reading that was sent only to the user, not to some group. Assume the User Model does not contains a plan for this goal, but that there is a plan for *compose.message*.

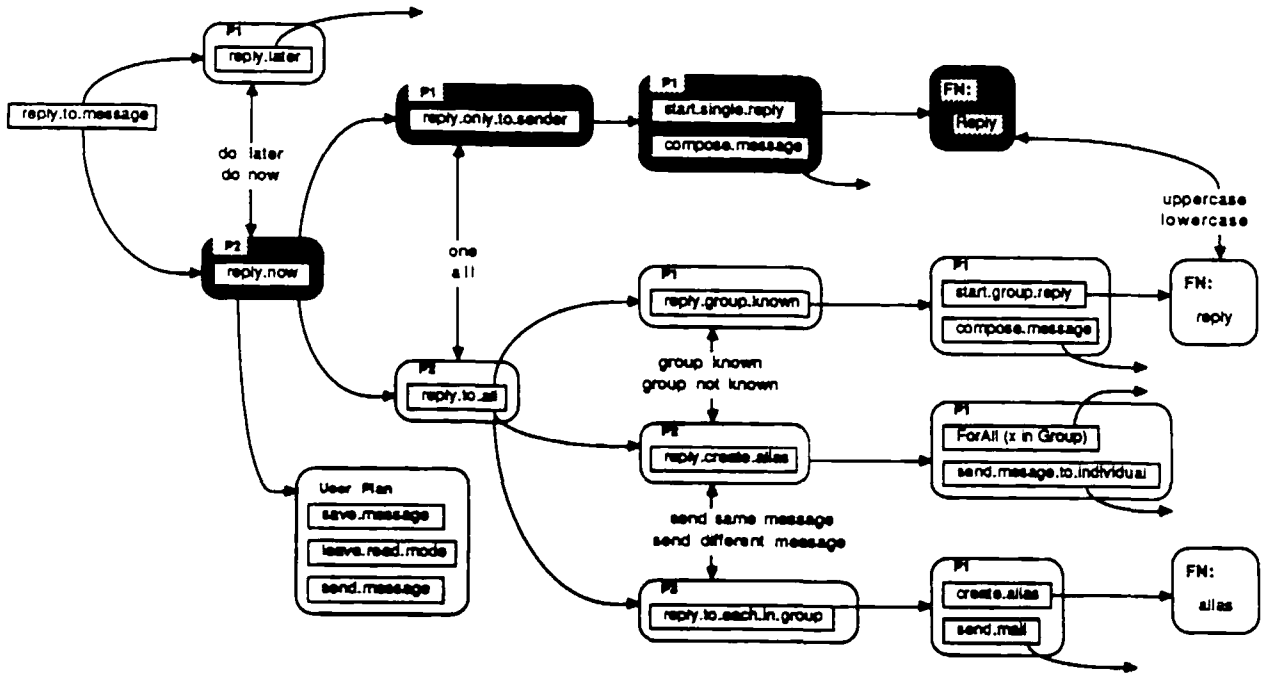

Figure 4-1: GENIE's Expert Model Knowledge of Replying to a Message

The PA starts searching at *reply.to.message*, a goal that has two alternative plans. Using the relationships between plans (called the choice set) it compares the single relation *do.now.later* to the Current Context. Since the user is engaged in reading mail; a rule for temporality chooses plan 2, which has a single sub-goal

reply.now. This is the only reason for choosing plan 2, so the PA checks the User Model, but finds no relevant knowledge that will help. The process is repeated on the sub-goal *reply.now* and because there is a relationship *all.one* and a single receiver in the Current Context, it chooses plan P1, having the single sub-goal *reply.only.to.sender*. This goal produces a single multi-step plan that satisfies it, and each of its steps are explored in turn. Once the best plan has been found, the User Model is traversed in order to find U. In this case the only component of B in U is *compose.message*. The Explainer chooses to *Introduce* because the user does not have a valid plan for the goal. Figure 4-2 presents both a trace of the PA and the template based text produced by the Explainer.

```
FINDBESTPLAN:
Construct plan for reply.to.message
  Looking for plan... choice set found: P1 P2
    Choosing by relations: do.now.later
      Checking World Knowledge...
        Choose P2: temporality: now BECAUSE user-engaged(reading)
      Checking User Model... Punt level reached
  CHOOSING P2
    Construct plan for reply.now
      Looking for plan... choice set found: P1 P2
        Choosing by relations: all.one
          Checking World Knowledge...
            Choose P1: groupsize: one BECAUSE one.receiver(user)
          Checking User Model...
            Choose P1: plan component known: compose.message
    CHOOSING P1
      Construct plan for reply.only.to.sender
        Looking for plan... single plan found: P1
          Construct plan for start.single.reply
            Looking for plan... function found: Reply
          Construct plan for compose.message
            Looking for plan...
                        (etc.)

Best Plan constructed for reply.to.message
*****

In order to reply to a message it is assumed you want to reply
right away.  In order to reply right away it is assumed you want
want to reply only to the sender.  To do this, you must indicate
you wish to reply and compose a message.  You can indicate you
wish to reply by using the command 'Reply'.

The syntax is: Reply
To use this command you must be reading
mail.  You temporarily stop reading mail; you start sending
mail; the receiver of your message is the current sender. To
compose a message just type your message and end with <esc>.
For example, Type the command: Reply
Then just type your message and end with <esc>.
```

Figure 4-2: An Example Finding the Best Plan

A second example illustrates how the choice of a better way is dependent upon the context and what the user knows. In one case the context includes a message sent only to the user in the other case to a group including the user. Consider the question "To reply to a group of users I reply to each individually — is there a better way?". The question identifies a goal — *reply.to.all* and a plan (S) — *Forall (x in group) send.message.to.individual.* It has a question intent (QI) to get the best plan. Assume that the User Model contains the goals, *compose.message* and *send.mail*, but does not contain *create.alias* or *start.group.reply.*

For the first case, assume the Current Context contains a message that was addressed to a group, and the User Model also contains *start.single.reply.* The top of Figure 4-3 presents a portion of the trace. All three plans are potential candidates, however a simple metric produces P3 as the winner. which has a single sub-goal *reply.group.known.* It is best because the group is known, it is less work , and the user knows one of its sub-goals. Since QI = get a better plan B than S, and S <> B, the Explainer chooses to *Clarify Responsively.*

In the second case, assume the Current Context contains a message sent only to the user. The bottom of Figure 4-3 presents a portion of the trace. The PA determines that the best plan is *reply.create.alias* since no group message exists, it is less work than the user's plan, and the user knows a sub-goal *send.mail.* Since QI = get a better plan B than S, and S <> B, the Explainer still chooses to *Clarify Responsively.*

5. Discussion

Since our focus is on content generation, we have not described how understanding occurs, nor how the knowledge representation is built or maintained. We are building a simple understanding component using current technology. At the present time, both the Expert and User models are built by hand. Developing the requisite knowledge acquisition and learning techniques would divert us from the generation task. Similarly, the Current Context is built by hand, although the monitoring techniques required for automation are being developed by Kaiser *et al.* [5]. Our attempt to use templates to produce text proved inadequate. The text tends to be silted and redundant, therefore we are exploring the use of a Functional Unification Grammar [6, 8]. Finally, GENIE currently only searches for known plans. We are confident that when failures occur, state space planning techniques can be employed.

To summarize, this paper has presented a frame-based approach to choosing alternative plans for question answering in a task-centered domain. We have shown how a Plan Analyst algorithm uses two sets of heuristics to traverse a representation of goals with alternative plans to choose a plan that is best in the current context and based on what the user knows. More importantly, we are able to abandon the use of stereotypes of functionality and expertise, which allows us to generate answers more relevant to users' needs.

```
FINDBESTPLAN:
  Construct plan for reply.to.all
    Looking for plan... choice set found: P1 P2 P3
      Choosing by relations: group.known.not same.different less.more.steps
        Checking World Knowledge...
        Choose P1: identified: group BECAUSE multiple.receiver(receiver.list)
        Choose P1: effort: less BECAUSE always good
        Choose P3: effort: less BECAUSE always good
      Checking User Model...
        Choose P1: known: compose.message -1
        Choose P2: known: reply.to.each.in.group 0
        Choose P3: known: send.mail -1
  CHOOSING P1
    Construct plan for reply.group.known
      Looking for plan... single plan found: P1
        Construct plan for start.group.reply
          Looking for plan... function found: reply
        Construct plan for compose.message
          Looking for plan... (etc.)
Best Plan constructed for reply.to.all
```

```
FINDBESTPLAN:
Construct plan for reply.to.all
  Looking for plan... choice set found: P1 P2 P3
    Choosing by relations: group.known.not same.different less.more.steps
      Checking World Knowledge...
        Choose P3: missing: group  BECAUSE single.receiver(user)
        Choose P1: effort: less BECAUSE always good
        Choose P3: effort: less BECAUSE always good
      Checking User Model...
        Choose P1: known: compose.message -1
        Choose P2: known: reply.to.each.in.group 0
        Choose P3: known: send.mail -1
  CHOOSING P3
    Construct plan for reply.group.known
      Looking for plan... single plan found: P1
        Construct plan for start.group.reply
          Looking for plan... function found: reply
        Construct plan for compose.message
          Looking for plan... (etc)
Best Plan constructed for reply.to.all
*****
```

Figure 4-3: Two Different Examples for Finding a Better Way

Acknowledgments

Special thanks to Kathy McKeown who made valuable comments on earlier versions of this paper. This work also benefitted from numerous conversations with Gail Kaiser, Elaine Kant and Cecile Paris.

References

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

2. Appelt, D. E.. *Planning Natural Language Utterances*. Cambridge University Press, Cambridge, England, 1985.

3. Chin, D. N. *Intelligent Agents as a Basis for Natural Language Interfaces*. Ph.D. Th., University of California, Berkeley, 1988.

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

5. Kaiser G. E., P. H. Feiler and S. S. Popovich. "Intelligent Assistance for Software Development and Maintenance". *IEEE Software* (May 1988), 40-49.

6. Kay, M. Functional Grammar. Proceedings of the 5th meeting of the Berkeley Linguistics Society, 1979.

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

8. McKeown, K.R. and Elhadad, M. Comparison of Surface Language Generators: A Case Study in Choice of Connectives. Presented at the Fourth International Workshop on Natural Language Generation, 1988.

9. Paris, C. L. *The Use of Explicit User Models in Text Generation: Tailoring to a User's Level of Expertise*. Ph.D. Th., Columbia University, 1987.

10. Pollack, M. *Inferring domain plans in question-answering*. Ph.D. Th., Moore School, University of Pennsylvania, May 1986.

11. van Beek, P. . A Model For Generating Better Explanations. Proceedings of the 25th Annual Meeting of the ACL, Association of Computational Linguistics, Palo Alto, California, 1987, pp. 215-220.

12. Wolz, U. Analyzing user plans to produce informative responses by a programmer's consultant. Tech. Rept. CUCS-218-85, Department of Computer Science, Columbia University, New York, NY, 1985.

13. Wolz, U. and G.E. Kaiser. A Discourse-Based Consultant for Interactive Environments. Proceedings of the Fourth IEEE Conference on Artificial Intelligence Applications, 1988, pp. 28 - 33.

14. Wolz, U., K.R. McKeown and G. E. Kaiser. Automated tutoring in interactive environments: A task centered approach. Tech. Rept. CUCS-392-88, Department of Computer Science, Columbia University, New York, NY, 1988.