An Object Oriented Approach to Content Planning for Text Generation

Ursula Wolz*

Columbia University
Department of Computer Science
New York, NY 10027

Technical Report: CUCS-004-90

February 28, 1990

Abstract

This paper describes GENIE, an object-oriented architecture that generates text with the intent of extending user expertise in interactive environments. Instead of generating text based solely on either discourse goals, intentions, or the domain, we found a need to combine techniques from each. We have developed an object oriented architecture in which the concepts about which we talk (domain entities), the goals that may be accomplished with them (intentions), and the rhetorical acts through which we express them (discourse goals) are represented as objects with localized knowledge and methods. A three stage process of content selection, content structuring and content filtering is presented. Content selection and structuring allow us to produce text that is within the context of the task at hand for the user. Content filtering allows us to revise and restructure the utterance to achieve clarity and conciseness.

Topic: Natural Language Generation
Keywords: Text Generation, Content Planning, User Modeling, Intelligent Interfaces

*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

A practical problem for text generation is how to produce good advice for users of interactive environments. In such settings users attempt to accomplish tasks by combining the set of available commands in a potentially infinite number of ways. Although canned resources may provide definitions or tutorials on how particular commands work, they cannot provide information within the context of the task at hand. To further complicate matters, users rarely exhibit classic patterns of expertise, but approach such environments with very particular task specific goals. Since users' primary goal is to complete the task, and only secondarily to get new information, they would prefer succinct, informative responses to queries during the *first shot*, rather than being diverted by an extended dialogue.

The GENIE project at Columbia has focused on how to build such an advice giving system. Instead of generating text based solely on either discourse goals [McKeown 85, Paris 87, Hovy 88, Moore & Swartout 89], on intentions [Appelt 85], or on the domain [Dale 88], we found a need to combine techniques from each. We have developed an object-oriented architecture in which the concepts about which we talk (domain entities), the goals that may be accomplished with them (intentions), and the rhetorical acts through which we express them (discourse goals) are represented as objects with localized knowledge and methods. Instantiated as objects, concepts such as goals, plans, actions and domain objects contain knowledge of how they interact with each other, and how they can be expressed textually. Similarly, rhetorical entities such as stating, comparing, elaborating can include knowledge of how they interact with each other, and how they may be applied to particular concepts or relations between concepts.

The GENIE [Wolz *et al.* 90, Wolz 90a] project has two major goals. First, the selection of content must be based on the current context of the user, which includes not only the problem with which the user needs help, but also the situation in which the help is requested, and the past experience of the user. Second, in any response, GENIE has two discourse goals. It responds informatively about the problem, and then attempts to extend the user's expertise by providing information as enrichment.

The three components of context introduce significant complexity into choosing what plan of action to suggest to the user. In particular they discourage the use of a cross product of stereotypical functionality and user expertise [Chin 88]. Instead they require the ability to both reason about the relationship between equally good plans, and explain that reasoning process. Context also influences the structure of the text. For example, a user who is familiar with a concept need only be reminded about it, and does not require a long introduction. Therefore, like other text generation systems, GENIE must first do *content selection* and choose which among a set of equally good plans should be described to the user. Then GENIE must do *content structuring*, casting the response in a form that most directly addresses the contextual needs of the user. The intrinsic modularity of an object-oriented approach helps to manage the complexity within each of these processes, and provides clean mechanisms for moving between them.

The dual discourse goals of responding and enriching introduce their own complexity, especially if they are to be combined into a single informative response. The purpose of enrichment is to sneak in a little extra information without distracting the user from the task at hand. Therefore, attempting to engage the user in a dialogue, while appropriate in other settings, is inappropriate here. The content necessary for satisfying the dual goals is often more than a user might tolerate in a single response however. Furthermore, many of the propositions that are included are redundant. These problems point to the introduction of a final phase of content planning, namely *content filtering*. Unlike current text planners that operate in a linear top-down fashion, GENIE is able to rewrite, or revise its response at the conceptual level, merging and restructuring the propositions that will eventually be manifested as text. This is possible because those propositions, which we call *rhetorical acts* are objects with knowledge about how they may and may not be integrated with each other.

The following section presents the motivation for our approach to text planning. Section 3 presents the architecture of GENIE. Section 4 compares this approach to related research.

Section 5 discusses the status of the implementation. Section 6 concludes with limitations and extensions.

2. Background

The GENIE project at Columbia [Wolz *et al.* 90, Wolz 90b, Wolz & Kaiser 88] addresses the problem of how to extend users' expertise in interactive environments. A communicative problem arises in such environments because users tend to get stuck in a *starter set* [Finin 83] of commands and never progress to more sophisticated methods for accomplishing tasks. The domain we consider is Berkeley Unix Mail, chosen because it is a microcosm of Unix, providing extremely rich and sophisticated functionality with a frustratingly hard to learn interface. The domain also allows us to focus on extending system expertise rather than domain problem solving since the activities, sending, receiving and managing messages, do not require high level thinking skills.

Interactive environments can be characterized as computational tools that allow users to accomplish tasks. Examples range from desk top aides such as word processors, message systems and spread sheets, to cad/cam systems and multi-purpose programming environments. Typically a core set of functionality can be combined and manipulated to construct solutions to domain specific problems. A user approaches such an environment with a domain specific or *computational* goal, and constructs a plan to satisfy that goal with the functional mechanisms available in the environment.

In such open-ended settings no two users will approach the system with the same background or needs. Consequently the degree of difficulty of functionality is more dependent upon the particular experience of a user rather than on broad categories of functional difficulty or user expertise. Users tend to develop their own highly personal repertory of commands. For example, an "expert" user who has extensive experience reading messages, and a "novice", one who has almost no experience reading messages, may both need to be introduced to a method for storing mail in files. The level of expertise of the first user is less significant than whether his or her particular expertise contributes to how much more easily the new method can be communicated to him or her. Furthermore, a particular goal may be satisfied by more than one plan, were the criteria for which plan is "best" is dependent up on the current context and the user's past experience.

Under these circumstances a pre-specified curriculum for learning about the environment, such as a text-based tutorial or hypermedia, is inappropriate. Even the best of tutoring systems [Clancey 82, Bonar *et al.* 86, Anderson *et al.* 85] will have expectations about the sequence in which particular skills are learned. A better approach is to model a human consultant or "guru" whom the user approaches to answer questions within the context of the current task in relation to what the user already knows. This type of behavior however requires an extremely flexible dynamic content planner. Without it, the information may not be sufficiently customized to the user's needs, causing confusion, rather than providing information. In GENIE, question answering is viewed as a three stage process of understanding the user's question, analyzing the pertinent information to select a course of action to suggest, and formulating that answer in a manner appropriate to the user's background. We concentrate on the second two stages with the following goals:

1. Provide information about the relationship between users' task (their computational goal) and the methods (plans) that may be used to accomplish them. Do this within the current situation in relation to what the user has done in the past. Respond informatively, but also attempt to provide enriching material.
2. Provide as much information in the first response so that the need for follow up questions is reduced, but provide it in a manner that is concise, clear and cohesive.
3. Explicitly include any contextual information that may not be obvious to the user.

These goals emerged as a result of two informal studies. First a set of textual materials, including manuals, tutorials, texts and canned text on-line resources were studied. Tutorials and on-line resources tended to focus on responding directly the problem of how to do tasks. Reference manuals, text books and especially "advanced user" manuals focused on enrichment. In all, 20 passages covering information about Unix, Lisp, Pascal, Logo and a number of word processors were analyzed. These texts have strongly influenced our approach to content selection and structuring. In particular, we found that four types of strategies occurred through which information about the relationship between plans and goals was given:

- **Introducing:** Presenting plans that the user has not encountered before.
- **Reminding:** Briefly describing plans to which the user has been exposed, but may have forgotten.
- **Clarifying Distinctions:** Explaining distinctions and options about plans.
- **Elucidating Misconceptions:** Clearing up misunderstandings that have developed about plans to which the user has been exposed.

The second analysis looked at 30 unix related question answering sessions in which the correspondence occurred through electronic mail. There were only two incidents of follow up questions. We found that the responses still fell within the four strategies. The respondent made an attempt to respond informatively and in 8 cases explicitly included contextual information that might not have been obvious to the questioner. In 7 cases the respondent included information that was not merely in response to the question, but that could extend the questioner’s expertise. Most importantly however, we found that many rhetorical strategies were often combined into very short utterances. The intent seemed to be to convey as much information as succinctly as possible.

3. Architecture of GENIE

Since the focus of research in GENIE is on text generation, we take the relevant aspects of context as input. How these may be derived is discussed briefly in section 6. GENIE also assumes a rather rich domain model that includes knowledge of intention and causality. Due to the nature of interactive environments, the primary task of GENIE is to provide information about how commands may be combined into plans to satisfy computational goals. This is to be contrasted with systems that explain the attributes of objects [McKeown 85], those that describe the causal connections between entities [Paris 87], and those that explain the actions of an agent such as an expert system [Moore & Swartout 89].

Given a particular user context, GENIE progresses through the three stages of text planning introduced in section 1. It first *selects content*, instantiating the computational goal — **G** under discussion, and relating it to a set of relevant plans. The plans are:

- **B** — the "best" plan, the plan that may be most appropriate for the goal.
- **S** — the stated plan, the plan to which reference is made in the question.
- **U** — the User Model plan, the plan GENIE believes the user has used in the past to satisfy the goal.
- **R** — the related plan, the plan that can be used to provide enrichment.

Once the relationships between the goal and the plans, and between the plans themselves has been established, GENIE begins *content structuring* in which one of the four strategies described above is selected for responding, and one for enriching. These strategies are then expanded until a complete set of relevant rhetorical acts is produced. Finally, GENIE *filters content* by applying rules that describe how the rhetorical acts may be *merged*, where they must be explicitly kept *separate*, when they may be *excluded* and when they must be explicitly *included*. GENIE employs three overlapping classes of objects; domain objects, intentional objects and discourse objects. *Domain Objects* include classifications of domain entities, including commands, and the entities manipulated by commands — within the Mail domain these include files, messages and users. *Intentional Objects* allow GENIE to construct and analyze plans, and include computational goals, plans, actions (which contain commands), effects and preconditions. *Discourse Objects* include rhetorical strategies, speech acts, syntactic structures and vocabulary. This section will show how once a current context is established, these objects are used for content selection, structuring and filtering.

3.1. Contextual Knowledge: Input to GENIE

In order for the response to occur within the current context we have identified the need for a three part user model. *A situational context* provides information on what activity the user is currently engaged in. This information may not be obvious to the user. *A discourse context* provides details about the user's elocutionary goal in asking the question. It contains information obvious to the user. *A model of user domain knowledge* that is a set of beliefs about a user's knowledge of commands and plans for combining those commands to satisfy goals. Because it is a set of beliefs of GENIE, it may contain information not obvious to the user.

The situational context is essentially a representation of the state in which the user asks a question. It may include details of where the user is, for example, in a unix shell rather than in the mail program. The discourse context is based on identifying the user's question as one of the following five types, and may include an instantiation of the goal and stated plan, if they are stated:

1. What plan will satisfy the goal? Given G, find B. Intuitively "How do I do it?"

2. What goal is satisfied by this plan? Given S, find G. Intuitively "What does it do?"

3. Does this plan satisfy this goal? Given S and G, confirm that S satisfies G. Intuitively "Does it do it?"

4. Why doesn't this plan satisfy this goal? Given S and G, explain why S does not satisfy G. Intuitively "Why doesn't it do it?"

5. Is there a better plan for this goal? Given S and G, find B. Intuitively "Is there a better way to do it?"

The model of user domain knowledge is an overlay to GENIE's own "expert" knowledge of intention and causality within the domain. The user model can however include knowledge that is not part of the expert model, and may contain faulty knowledge. More will be said about the expert model shortly. Consider the following scenario. A user is in the Unix environment, rather than in the mail program itself, and there is no new mail for the user. In the past the user has only sent mail after reading new mail, that is, only after entering the mail program to read new mail. The user has never used the option for entering the mail program for the purpose of sending a message, only for the purpose of reading mail. The user asks the question "How can I send a message?" The situational context includes the fact "user is in unix." The user model of domain knowledge would indicate no experience with entering the mail program for the purpose of sending mail. The discourse context includes a goal "send mail", in which a single message, rather than a group of messages is specified. It expects a plan for satisfying the goal and assumes such a plan exists. It does not, however specify to whom the message should be sent. Contrast this with the question "Why doesn't "mail McKeown" let me send a message to Kathy?" Here both a stated plan and a goal are provided along with a specification of the recipient. However, the assumption is that the plan is faulty and the expectation is that the response will explain why.

GENIE's hierarchical object base of domain knowledge includes knowledge of *intentions* — how to accomplish domain specific tasks, *causality* — what the results of actions are, and *system operations* — the core functionality on which the entire system is based. The intentional layer encodes knowledge of goals and how to satisfy them through actions. Goal satisfaction can also be encoded as composite plans, which are collections of goals that in turn may be satisfied through actions or composite plans. For explanation purposes, we also saw the need for abstracting the knowledge typically encoded in preconditions. Often, the explanation for why an action should be taken requires knowing semantic information about the relationship between preconditions, rather than merely that a precondition has been met. For example, in the mail domain one can send a message from the Unix environment, or from inside the mail environment. By encoding this knowledge as a *discriminator* object with type "mode" and with features "unix" and "mail", we still have the power of choice of a classic precondition, but have the added explanatory power to be able to relate the chosen feature to the feature that failed.

In GENIE, the action layer is represented in a more classic manner. Actions have preconditions, which when met, spawn effects. Effects may be represented as composites — leading to collections of other actions, or as system level operations. Again, for explanatory purposes, a precondition contains added information: the goal that can satisfy it, and the effect that occurs if the action is invoked when the precondition is not met. For example, one can attempt to read a file of mail messages, but if the file doesn't exist, the action causes an error message to appear which is a predictable, though unintended effect. Figure 3-1 shows a portion of the knowledge base required for the question "How can I send a message?" The model of user domain knowledge is an overlay to this knowledge, as can be seen from the Figure.

Figure 3-1: Partial Map of the Intentional Objects for the Send Mail Goal

Each of these classes of objects contain semantic and lexical information. For example, a goal can be expressed as a *process action* which has semantic components agent, process-verb, medium, beneficiary and/or location attributes. The generic goal "send mail" has uninstantiated agent and beneficiary, a process-verb "send", and medium "mail." The semantic concepts "send" and "mail" have lexical attachment to particular verbs and nouns. In particular, since "mail" is also a domain concept it may be instantiated as a subclass with more semantic detail. This is illustrated more naturally through reference to users. Consider the difference between the question "How do I a message to Kathy" and "How do I send a message." In the first sentence, the goal "send mail" is related to the object "Kathy" which includes the fact that reference to the user should be through the proper name "Kathy." Instantiating the second goal requires instantiating a *default* beneficiary relation, which, as will be seen in our example include semantics for producing either the phrase "an individual", or "individuals."

3.2. Content Selection: Objects Reason About Themselves

Once the contextual knowledge from the situation and the discourse have been established as instantiated objects, GENIE is ready to select the content of the response. This requires instantiating the goal and the four plans listed in section 3.1, and comparing these objects. Each of the objects is sent two messages: construct-yourself and compare-yourself. The execution is dependent upon the class of the object. The messages may be sent in any order. A blackboard keeps track of which objects have returned successful constructs and compares, and messages are resent until all objects return successfully. The methods are sophisticated enough to prevent redundancies. For example, if while executing its compare, B successfully compares itself with S and U, then S and U are all marked as compared to B. When S is asked to compare itself, it need only compare itself with U, as it already knows its relation to B.

Constructing proceeds as follows. If the goal G has not been constructed in the discourse context, (if it wasn't included in the question), then its method traverses the domain knowledge of intention in an action to goal direction, using the actions from the stated plan S. The best plan B traverses the domain knowledge of intention in a goal to action direction, using G. Both use the discriminators to guide search based on five ordered heuristics:

1. If knowledge from the discourse context doesn't conflict with knowledge from the situational context, rely on the discourse context.
2. Rely on the situational context.
3. Rely on the model of user knowledge.
4. Rely on default heuristics for the domain entity that corresponds to the type of the discriminator.
5. Don't make a decision.

If the plan S is not present from the discourse context, it merely marks itself as absent. The user model plan is constructed in a manner similar to the best plan, but is restricted to knowledge that is marked as known by the user. The related plan R is dominated by B, that is, B sends R "close calls" at decision points, giving R the branch that lost. Comparison between G and a plan requires validating that the plan is works within the current situational context by applying a simulation technique to the causal layer of domain knowledge. If the plan fails, the explanation for failure is recorded. Comparing plans with other plans uses a simple matching algorithm. The objects are annotated as to whether they are equivalent, or if differences occur, where those differences are. This information is critical to answering "what went wrong" and "what is a better way" questions.

Returning to the send mail example, only a goal is given. Therefore, S is marked as absent, B, U and R are constructed and compared. Given the input context, B is instantiated as a plan to "send from unix" with "recipient is individual." The first decision is based on heuristic 2, the situation. The second decision is based on heuristic 3, since the user model only contains experience with individuals as recipients. When comparison occurs, U discovers that it is not valid in the current context because one of the preconditions to one of its commands is missing — there is no new mail. When content selection is finished, in this example, over 100 objects are instantiated including goals and subgoals, discriminators, actions, subactions, preconditions, and effects. Clearly not all of them should be included in the response.

3.3. Structuring Content: Instantiating Rhetorical Objects

The result of an analysis may produce significantly more information than is necessary to help the user. For example, the details of the best plan may be extensive, but the model of the user's knowledge reveals that the user knows most of it already. From the analysis of text, we identified four strategies or high level themes through which task information is presented. These are the strategies listed in section 2. Each strategy can be used either to respond or to provide enrichment.

The strategies contain explicit structure [Wolz *et al.* 90] that is discourse rather than domain dependent. This phenomenon is well established, having been first identified by McKeown [McKeown 85]. In GENIE we represent these structures as rhetorical objects that can be instantiated, and can refer and send messages to one another and to domain and reasoning objects. In this way the five objects that form the content can influence the structure of the response, and independent rhetorical entities can influence content.

Once the content selection stage is complete, each of the plans (B, S, U and R) is sent a message "talk-about-yourself." Depending on the type of the plan (best, user model, etc.) this message will select two high level rhetorical strategies. The decision is based on a compilation of rules derived from the design goals. Examples of the rules are shown in Figure 3-2. In the "how do I send mail" example, B selects "IntroRespond" because there is no stated plan, and the user model plan is invalid. U selects "ElucidateEnrich" because it is not valid for G. R remains silent because U is not equal to B. S remains silent because it is absent. Having selected a rhetorical strategy, each of the plans creates an instance of its strategies, links itself to the instantiation and sends a message to the strategies to expand themselves. The strategies contain rhetorical plans for expanding themselves. The rhetorical plans may either expand into other rhetorical strategies or terminate in *rhetorical acts*. The top level strategy for Introducing, that is called by IntroRespond is shown in Figure 3-3 along with the first level expansion into other strategies and acts. Plan expansion occurs by sending messages to components of the plan until the plan bottoms out as a set of rhetorical acts.

Notation:	P_E = P exists $P_{\sim E}$ = P does not exist
	P_V = P is valid $P_{\sim V}$ = P is not valid
	& = AND, | = OR
TalkAbout: B	If $S_{\sim E}$ & ($U_{\sim E}$ | $U_{\sim V}$) IntroRespond
TalkAbout: U	If $S_{\sim E}$ & U_V RemindRespond
	If $S_{\sim E}$ & (U_V <> B) ClarifyEnrich
	If $U_{\sim V}$ & ($S_{\sim E}$ | [S_V = B]) ElucidEnrich
TalkAbout: R	If [$S_{\sim E}$ | S_V = B] & (U_V = B) IntroEnrich

Figure 3-2: Summary of Conditions that Determine the Strategy

Rhetorical acts contain functional descriptions which when given to the surface generator can produce text. The form of these descriptions is based on the theory of functional grammar [Halliday 85]. They also include knowledge for how they interact with other rhetorical acts, as will be seen in the next section. The execution of rhetorical strategies may produce redundant, extraneous or obvious information. For example, Figure 3-4 shows the text that would be produced if the first 6 rhetorical acts instantiated in the send mail example were given to the surface generator. In all 30 rhetorical acts are produced because of the expansion of subactions of send mail command.

```
(rh-plan introduce composite-plan           ;to introduce a composite-plan send
    ((state-it goal-satisfied required)     ;state-it message to goal-satisfied attribute,
     (describe-decisions choice-made self)  ;send self describe-decisions using list
      (summarize subgoals required self)    ;in choice-made attribute, etc.
      (expand subgoals required self)))

BEST-PLAN sending INTRODUCE message to {plan} PL-SEND-MAIL-2
PL-SEND-MAIL-2 sending STATE-IT message to {goal} SEND-MAIL-1
STATE-IT-GOAL-1 {r-act} instantiated
PL-SEND-MAIL-2 sending DESCRIBE-DECISIONS message to {plan} PL-SEND-MAIL-2
PL-SEND-MAIL-2 sending STATE-IT message to {discriminator} SEND-MAIL-MODE-1
SEND-MAIL-MODE-1 sending STATE-IT-STRONG message to {discriminator} SEND-MAIL-MODE-1
STATE-IT-STRONG-DISCRIMINATOR-1 {r-act} instantiated
PL-SEND-MAIL-2 sending EXPAND message to {plan} PL-SEND-MAIL-2
PL-SEND-MAIL-2 sending INTRODUCE message to {action} SEND-FROM-UNIX-1
```

Figure 3-3: Rhetorical Strategies For Introducing

```
STATE-IT-GOAL-1 :: DOMAIN-OBJECT[GOAL]:SEND-MAIL-1
STATE-IT-STRONG-DISCRIMINATOR-1 ::
  DOMAIN-OBJECT[DISCRIMINATOR]:SEND-MAIL-MODE-1
STATE-IT-GOAL-2 :: DOMAIN-OBJECT[GOAL]:CHOOSE-RECIPIENTS-1
STATE-IT-WEAK--DISCRIMINATOR-1 ::
  DOMAIN-OBJECT[DISCRIMINATOR]:CHOOSE-RECIPIENTS-RECIPIENT-1
STATE-IT-GOAL-3 :: DOMAIN-OBJECT[GOAL]:SEND-INDIVIDUAL-1
STATE-IT-WEAK-DISCRIMINATOR-2 ::
  DOMAIN-OBJECT[DISCRIMINATOR]:SEND-INDIVIDUAL-ADDRESS-1

You want to send mail.
I know you are in Unix.
You want to specify a recipient.
I assume your recipient is an individual.
You want to specify a single recipient.
I assume the address is a local address.
```

Figure 3-4: Example of Text Produced Before Content Filtering

3.4. Filtering Content: Communication Between Rhetorical Acts

The structuring stage produces an initial ordering of rhetorical acts. The filtering stage reviews this list and specifically includes, excludes, merges or keeps separate acts within this list. For example information derived from either the situational context, such as the mode the user is in, should be included. Information derived from the discourse context, such as an explicit reference to the goal may be excluded as obvious. In the process of both reminding and clarifying, the same preconditions may be introduced twice. These can be merged into one utterance. However, if a component of an utterance is critical, such as a failed precondition when elucidating a misconception, then it should be kept separate. Restructuring occurs when recursive merges and excludes occur. A merge may move an act up or down depending on whether the dominant act comes before or after the submissive one. Two acts which are initially separated by some distance may find themselves next to each other, or even merged, if the acts between them are excluded.

The rhetorical acts themselves contain methods for determining how they can be filtered. This includes specific rules for when they may be excluded, when they must be included, what other objects they may or must be merged with and which one is dominant, and when they are to be kept separate.

The final stage of content planning occurs when the set of rhetorical acts is sent messages to "execute" themselves. Each examines its methods for how it may be merged etc. Like the message passing between the content plans, processing continues until all objects return successfully. In the process some objects will have been excluded and some will have instantiated new "merged" objects, subordinating themselves to the merged object. Once processing is complete, the remaining objects apply knowledge for constructing clause level structures which are passed to the surface generator.

The resulting text for the send mail example appears in Figure 3-5. Figure 3-6 shows how the 6 acts from Figure 3-4 are filtered to produce the first sentence in figure 3-5. The first act of stating the main goal can be excluded because it is in the discourse, and is probably obvious to the user. The second and third goals may be excluded because they are in the chain of objects from the primary goal to an action in which there is never more than one step for any subgoal. Intermediate STATE-IT-GOALs are only kept if there is some problem with the plan for that goal, or it contains more than one step. Similarly, the justification for decisions that are manifested through DESCRIBE-DISCRIMINATORs strategies can be merged when such goal, plan, goal chains occur. Finally, STATE-IT-WEAK-DISCRIMINATORs and STATE-IT-STRONG-DISCRIMINATORs can be merged when the number of dominant ones (weak) out number the submissive ones (strong). The "strong" act uses vocabulary such as "Since", while the "weak" uses vocabulary such as "Asssuming."

```
I assume you are in unix, your recipient is an individual, and the
address is a local address. You must supply the individual's email
address. For example, in order to send mail to Kathy, type "mail
kathy", since Kathy is her email address.  Your usual method is to
read your mail, then use the mail command inside the mail environment.
Since you do not have new mail, your old method will not work.
```

Figure 3-5: Example of Text Produced After Content Filtering

```
Excluded! STATE-IT-GOAL-1 because IN-DISCOURSE-CONTEXT
Excluded! STATE-IT-GOAL-2 because IN-GOAL-PLAN-GOAL-PATH
Excluded! STATE-IT-GOAL-3 because IN-GOAL-PLAN-GOAL-PATH
Merged! STATE-IT-WEAK-DISCRIMINATOR-2 and STATE-IT-WEAK-DISCRIMINATOR-1
 into CONJOINED-WEAK-DISCRIMINATOR-1
 because IN-GOAL-PLAN-GOAL-PATH
Merged! CONJOINED-WEAK-DISCRIMINATOR-1 and STATE-IT-STRONG-DISCRIMINATOR-1
 into CONJOINED-WEAK-DISCRIMINATOR-2
 because IN-GOAL-PLAN-GOAL-PATH and WEAK dominates STRONG

I assume you are in unix, your recipient is an individual, and the
address is a local address.
```

Figure 3-6: Application of Filtering Methods

4. Related Work

The work in this paper extends and contrasts with other research in two aspects of text generation. First, in its perspective on content selection through context, and then in its ability to revise the content.

Our first distinction is in the emphasis on a *first shot* in a dialogue. This is in contrast to

Moore [Moore & Swartout 89] whose important contribution is the ability to handle follow up questions, while down playing the emphasis on the initial response. In particular, her first response does not include the background information about content planning decisions that we include. Although GENIE does not contain the methods for following up, the structures produced in content structure and filtering contain the same rhetorical planning information as Moore's system. Our approach to content selection must also be contrasted with work by Chin [Chin 88] in our emphasis on contextual rather than stereotypical knowledge. Unlike Chin we do not need rely on a cross product of functional and user stereotypes. Our argument is that users will almost never fall cleanly into any such stereotypes, and the application of them may only result in frustrating the user.

Finally our work is distinguished from text planning that relies on discourse goals in our ability to revise. Although our strategies bear striking similarity to Schemata [McKeown 85, Paris 87, McCoy 86], the modularity of strategies provides a larger degree of flexibility. Like schemata, structure is controlled by discourse knowledge. However strategies also allow structure to develop as a result of intentional and domain knowledge. Rhetorical Structure Theory (RTS) [Mann & Thompson 87], especially as it is applied to text planning [Moore & Paris 89, Hovy 88] provides a large degree of flexibility in choosing how to expand structure. However, it is not clear whether any discipline is applied to the expansion of rhetorical relations. More importantly, both Schemata and RTS proceed in a linear, top-down fashion, and do not provide for any sort of filtering or revision.

5. Status and Implementation

GENIE is implemented on a Sun 3/60 in Sun Common Lisp. All of the object types described are represented using Hyperclass [Smith and Carando 86] which provides powerful inheritance mechanisms. Surface text generation is accomplished through FUF [Elhadad 88] that employs a two stage process of functional unification [Kay 79] and linearization to produce English text. All of the functionality of Berkeley Unix Mail can be represented with the exception of command files which allow customization by loading directives into the mail environment.

The methods for plan analysis have been implemented as lisp functions for purposes of efficiency. The methods for content structuring have been implemented using the formalism shown in figure 3-3. Although we do not presently see a need to extend the formalism, it is a straightforward process to do so. The formalism currently supports both binary (if/then/else) and n-ary(case) branching, recursion, and iteration on a list. The methods for content filtering are currently implemented as lisp code. We are in the process of defining a formalism for these too. The formalisms themselves are an attempt to abstract out the salient features of these primarily linguistic processes in order to allow experimentation which might lead to more formal theories to support the mechanisms that are implemented. Work on integrating completely with FUF remains to be done. At the present time FD skeletons and vocabulary is only available for a limited set of examples.

Our approach provides a large degree of flexibility. For any single computational goal, at least 120 scenarios can be generated [Wolz 90b]. It remains to be seen whether this range significantly impacts the range of text that is generated. We are about to conduct an analysis of the relationship between the text produced and the input scenarios. We expect to verify statistically that the occurrence of specific contextual entities in the text can be predicted by the information included in the input. Furthermore we expect to show that across texts, the number of sentences produced is considerably smaller than the number of such contextual entities.

6. Limitations and Conclusions

Two open questions remain. First, can the user model required as input to GENIE be produced automatically. Second could GENIE be extended to enable it to engage in an extended dialogue. Building and maintaining the user model would involve developing a system that can observe user behaviors and draw inferences about user knowledge from those behaviors. There are two obvious observable behaviors, the questions asked by the user, and the actions taken that affect the situation. Part of the solution to maintaining the user model may be found in extending the communicative ability of GENIE to allow it to engage in an extended dialogue. In order to do so, GENIE would need to be able to maintain a history of the discourse context, the situational context in which the preceding discourse took place, and the reasoning it used to choose a response. With such data, comparisons could be made about what entities were talked about or used over time, allowing conclusions about user knowledge to be drawn. Currently GENIE contains the requisite knowledge representations, we are confident that the appropriate knowledge acquisition systems can be built.

Another open question is the problem of mapping GENIE's rhetorical acts to surface forms. In the present implementation the functional descriptions embedded in the acts is extremely primitive. Although Fuf supports lexical choice, it is not presently exploited in GENIE, and although we make use of complex sentences, there is no principled theory for our choice of sentence structure. Work by Elhadad [Elhadad 90] is directed toward these problems.

To summarize, in this paper we have shown how knowledge of the domain, intentions and discourse can be combined in an object oriented architecture for text planning. A three stage process of content selection, content structuring and content filtering is employed by GENIE as it responds informatively to users and opportunistically enriches their expertise.

Acknowledgements

Kathy McKeown and Gail Kaiser deserve thanks for supporting this work. Dr. McKeown also provided valuable suggestions on an earlier version of this paper. I am grateful to David Robinowitz and Wonsuk Jang for proving that GENIE could be integrated with Fuf, and to Michael Tanenblatt for the readability of his code.

References

[Anderson *et al.* 85]
J.R. Anderson, C.F. Boyle and B.J. Reiser.
Intelligent tutoring systems.
Science 228(4698):456-462, April, 1985.

[Appelt 85] Appelt, D. E.
Planning Natural Language Utterances.
Cambridge University Press, Cambridge, England, 1985.

[Bonar *et al.* 86] J.G. Bonar, R. Cunnigham and J. Schultz.
An object-oriented architecture for intelligent tutoring.
In *Proceedings of OOPSLA '86*, pages 269-276. Association for Computing Machinery, Portland, OR, September, 1986.

[Chin 88] Chin, D. N.
Intelligent Agents as a Basis for Natural Language Interfaces.
PhD thesis, University of California, Berkeley, 1988.

[Clancey 82] Clancey, W.J.
Tutoring rules for guiding a case method dialogue.
Intelligent Tutoring Systems.
Academic Press, London, 1982, pages 201-225.

[Dale 88] Dale, R.
Generating referring expressions in a domain of objects and processes.
PhD thesis, University of Edinburgh, 1988.

[Elhadad 88] Elhadad, M.
The FUF Functional Unifier: User's manual.
Technical Report CUCS-408-88, Columbia University, June, 1988.

[Elhadad 90] Elhadad, M.
Constraint-Based Text Generation: Using Local Constaints and Argumentation to generate a turn in conversation.
Technical Report CUCS-003-90, Columbia University, New York, NY, 1990.

[Finin 83] Finin, T.
Providing help and advice in task oriented system.
In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 176-178. Karlsruhe, West Germany, 1983.

[Halliday 85] Halliday, M.A.K.
An Introduction to Functional Grammar.
Arnold, London, 1985.

[Hovy 88] Hovy E.H.
Two types of planning in language generation.
In *Proceedings of 26th Meeting of ACL*, pages 179 - 185. Association for Computational Linguistics, 1988.

[Kay 79] Kay, M.
Functional Grammar.
In *Proceedings of the 5th meeting of the Berkeley Linguistics Society*. Berkeley Linguistics Society, 1979.

[Mann & Thompson 87] W.C. Mann and S.A. Thompson.
Rhetorical Structure Theory: A Theory for Text organization.
The Structure of Discourse.
Ablex, Norwood, NJ, 1987.

[McCoy 86] McCoy, K. F.
The ROMPER System: Responding to Object-Related Misconceptions Using Perspective.
In *Proceedings of the 24th Annual Meeting of the ACL*. Association of Computational Linguistics, New York City, New York, June, 1986.

[McKeown 85] McKeown, K.R.
Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.
Cambridge University Press, Cambridge, England, 1985.

[Moore & Paris 89] Moore, J.D. and C.L. Paris.
Planning text for advisory dialogues.
In *Proceedings of 27th Meeting of ACL*, pages 203 - 211. Association for Computational Linguistics, 1989.

[Moore & Swartout 89] Moore, J. D. and Swartout W. R.
A Reactive Approach to Explanation.
Submitted to the: ‘‘Eleventh International Joint Conference on Artificial Intelligence’’ Detroit, Michigan.
August, 1989
also presented at the Fourth International Workshop on Natural Language Generation on July 1988.

[Paris 87] Paris, C. L.
The Use of Explicit User Models in Text Generation: Tailoring to a User’s Level of Expertise.
PhD thesis, Columbia University, 1987.

[Smith and Carando 86] Smith, R.G and P.J. Carando.
Structured Object Programming In Strobe.
Technical Report, Schlumberger-Doll Research, Ridgefield, CT, 1986.

[Wolz 90a] Wolz, U.
Extending User Expertise in Interactive Environments.
PhD thesis, Columbia University, 1990.
Forthcoming.

[Wolz 90b] Wolz, U.
The impact of user modeling on text generation in task-centered settings.
In *Second International Conference on User Modeling*. Honolulu, Hawaii, 1990.

[Wolz & Kaiser 88]
Wolz, U. and G.E. Kaiser.
An Automated Consultant for Interactive Environments.
Technical Report CUCS-391-88, Department of Computer Science, Columbia University, New York, NY, 1988.

[Wolz *et al.* 90] Wolz, U., K.R. McKeown and G. E. Kaiser.
Automated tutoring in interactive environments: A task centered approach.
Machine-Mediated Learning 3(1):53-79, 1990.