INDEXING AND EXPLOITING A DISCOURSE HISTORY TO GENERATE CONTEXT-SENSITIVE EXPLANATIONS

Johanna D. Moore[*]

Department of Computer Science, and LRDC
University of Pittsburgh
Pittsburgh, PA 15260

ABSTRACT

A striking difference between the interactions that students have with human tutors and those they have with computer-based instruction systems is that human tutors frequently refer to their own previous explanations. Based on a study of human-human instructional interactions, we are categorizing the uses of previous discourse and are developing a computational model of this behavior. In this paper, I describe the strategies we have implemented for identifying relevant prior explanations, and the mechanisms that enable our text planner to exploit the information stored in its discourse history in order to omit information that has previously been communicated, to point out similarities and differences between entities and situations, and to mark re-explanations in circumstances where they are deemed appropriate.

1. Introduction

To reap the benefits of natural language interaction, user interfaces must be endowed with the properties that make human natural language interaction so effective. One such property is that human speakers freely exploit all aspects of the mutually known context, including the previous discourse. Computer-generated utterances that do not draw on the previous discourse seem awkward, unnatural, or even incoherent. The effect of the discourse history is especially important in instructional applications because explanation is essentially incremental and interactive. To provide missing information in a way that facilitates understanding and learning, the system must have the capability to relate new information effectively to recently conveyed material, and to avoid repeating old material that would distract the student from what is new. Strategies for using the discourse history in generating utterances are therefore crucial for building computer systems intended to engage in instructional dialogues with their users.

The goal of our work is to produce a computational model of the effects of discourse context on explanations in instructional dialogues, and to implement this model in an intelligent explanation facility. Based on a study of human-human instructional dialogues, we are developing a taxonomy that classifies the types of contextual effects that occur in our data according to the explanatory functions they serve [1]. Thus far, we have focused on four categories from our taxonomy

- explicit reference to a previous explanation (or portion thereof) in order to point out similarities (differences) between the material currently being explained and material presented in earlier explanation(s),
- omission of previously explained material to avoid distracting student from what is new,
- explicit marking of repeated material to distinguish it from new material (e.g., "As I said before, . . . ")
- elaboration of previous material in the form of generalizations, more detail, or justifications.[1]

Building on previous work [2, 3] we have implemented an explanation facility that maintains a discourse history and uses it in planning subsequent explanations. We are using this explanation facility as part of two intelligent systems. The first is a patient education system intended to provide patients with information about their disease, possible therapies, and medications [4, 5]. The second is an intelligent coached practice environment for training avionics technicians to troubleshoot complex electronic equipment [6].

In order to generate texts that exploit previous discourse, a system must have the following capabilities:

1. It must understand its own previous explanations.
2. It must be able to find prior explanations (or parts thereof) that are relevant to generating the current explanation, *in an efficient manner.*
3. It must have strategies for exploiting the relevant prior texts in pedagogically useful ways when generating the current explanation.

In this paper, I describe how we have realized these three requirements in the two instructional systems.

2. Background

To achieve the first requirement, our explanation system uses an extended version of the text planner developed by Moore

[*]The research described in this paper was supported by the Office of Naval Research, Cognitive and Neural Sciences Division, and the National Science Foundation, Research Initiation Award.

[1]This category breaks up into number of subcategories in our taxonomy.

and Paris [2]. Briefly, the text planner works in the following way. When the user provides input to the system, the query analyzer interprets the question and forms a *communicative goal* representing the system's intended effect on the hearer's mental state, e.g., "achieve the state where the hearer believes that action A is suboptimal" or "achieve the state where the hearer knows about the side effects of drug X." A linear planner synthesizes responses to achieve these goals using a library of explanation operators that map communicative goals to linguistic resources (speech acts and rhetorical strategies) for achieving them. In general, there may be many operators available to achieve a given goal, and the planner has a set of *selection heuristics* for choosing among them. Planning is complete when all goals have been refined into speech acts, such as INFORM and RECOMMEND.

In this system, a text plan represents the effect that each part of the text is intended to have on the hearer's mental state, the linguistic strategies that were used to achieve these effects, and how the complete text achieves the overall communicative goal. When a text plan is complete, the system presents the explanation to the user, retaining the plan that produced it in a *discourse history.*

In previous work, I showed how a system could support a limited range of dialogue capabilities using the information recorded in its discourse history [3]. In particular, I devised interpretation and recovery heuristics that examine the text plan that produced the immediately preceding explanation in order to interpret and respond to the follow-up questions "Why?" and "Huh?". In addition, the system is able to avoid producing the same answer to a question asked a second time by searching the discourse history to determine if the communicative goal corresponding to the question was ever posted before. If so, the system notes which strategy was used in the previous case and employs recovery heuristics to choose an alternative strategy.

The work reported in this paper is aimed at augmenting the ways in which the information recorded in the discourse history affects each new explanation as it is planned. In general, previous dialogue should potentially influence the answer to *every* subsequent question, not just explicit follow-up questions, such as "Why?" and "Huh?", or questions that are literally asked twice. Supporting this capability requires recognizing when prior explanations are relevant and how they should affect the current response.

3. Examples

Examples of the types of contextual effects we are interested in appear in Figures 1 and 2. The dialogue in Figure 1 is taken from our corpus of human-human written instructional dialogues in the SHERLOCK domain. SHERLOCK is an intelligent training system that teaches avionics technicians to troubleshoot complex electronic equipment. It is built within the "learning by doing" paradigm, in which students solve problems with minimal tutor interaction and then review their troubleshooting behavior in a post-problem *reflective follow-up* session (RFU) where the tutor replays each student action and provides a critique (here the tutor marks each action as "good" (<+>) or as "could be improved" (<->)). To collect protocols for study, the system was used to replay each step during RFU, but the human tutor provided the assessment and answered any questions the student posed.

In Figure 1, the student performs three actions that are assessed negatively for reasons that are related. Testing pin 28 is bad for one of the same reasons as testing pin 38, and testing pin 36 is bad for precisely the same reason as testing pin 28. In the figure, italics are used to highlight what we categorize as contextual effects on the explanations given. For example, when explaining why testing pin 28 is bad (turn 6), the tutor refers back to one of the reasons given in the explanation in turn 3, and reiterates the fact that the main data inputs are highly suspect (signalled by "As explained before"). In turn 6, the tutor offers an elaboration that introduces the notions of main and secondary data control signals and justifies why the main data signal should be tested first. Later, when explaining why testing pin 36 is bad in turn 9, the tutor refers back to the explanation he gave for testing pin 28 and states a generalization explaining why all three of these actions are considered suboptimal. The tutor expects the student to be able to make use of the explanations given in turn 6 (and therefore turn 3) by indicating that it is relevant to the current situation ("for the same reasons given ..." serves this purpose). Accordingly, the tutor does not repeat the detailed explanation of why the main control data signals should be tested first. By generating the explanation in turn 9 in such a way that it 'meshes' with the previous two, not only does the tutor correct student's error, but he forces the student to consider how the three situations are similar. Pointing out this similarity may facilitate the student in forming the domain generalization and recognizing how the three instances fit this generalization.

Figure 2 shows an actual dialogue with our patient education system, and is based upon phenomena we observed in naturally occurring interviews with physicians. The responses shown in this example are automatically generated, and typify the type of interaction a patient can have with our system. Patients construct questions via a direct manipulation interface that allows users to combine menu selection and highlighting portions of previous explanations on the screen. This interface extends our previous work [7]. Note that although the user asks exactly the same type of question in turns 1 and 3, the system's answers are quite different. This is because the answer in turn 4 (R4) is affected by the context created in turns 1–3. For example, the first sentence of R4 points out that Elavil is used for the same type of therapy as Inderal. Further

TUTOR	<->You tested pin 38 of the A1A3A15.	[1]
STUDENT	Why is testing pin 38 not considered a good step?	[2]
TUTOR	For two reasons. First, because you did not completely test the inputs needed to set relay B24 (you tested the low input but not the high input). These two inputs are critical for the TO test step and are highly suspect when a relay appears to be reset. Secondly, the data input at pin 38 has no effect on the path that is being created between pins 57 & 58 and the output at 11 & 12. Relay B26 could be set or reset – it just doesn't matter.	[3]
⋮	***Short discussion about how to determine whether a relay interferes with the signal path.***	
TUTOR	<-> You tested pin 28 of the A1A3A15.	[4]
STUDENT	Why is testing pin 28 not a good step?	[5]
TUTOR	***As explained before,*** the main data inputs to relay B24 are highly suspect at this time. You have tested the low input to B24 but not the high one. ***Since it is more likely that there may be a problem with the main data signals to relay B24 than with the secondary data signals to relays B25-B30, the main data signal should be tested first.*** Testing pin 28 is not a bad step, it is just more likely (on the average) that the main data signals are causing the problem.	[6]
⋮	***One exchange and one action later.***	
TUTOR	<->You tested pin 36 of the A1A3A15.	[7]
STUDENT	Don't I need to test pin 36?	[8]
TUTOR	You might have to, but ***for the same reasons given when you tested pin 28, it is generally more efficient to test the main control data signals first, and then test the secondary control data signals if necessary.***	[9]

Figure 1: Human-Human Advisory Interaction Displaying Contextual Effects

note that in R4 the system does not explain what prophylactic treatment means because it has done so previously in R2, i.e, the system omits information that has been presented in a previous explanation. Finally, in the penultimate sentence of R4, Elavil's contraindications and side effects are contrasted with those of Inderal.

4. Finding Relevant Prior Explanations

In order to produce the types of behavior exemplified above, a system must be able to determine which prior explanation(s) should be referred to when constructing an explanation. Although it is convenient for expository purposes to think of the tasks of finding relevant prior explanations and exploiting them in the construction of the current explanation as conceptually distinct, they may be interleaved in the actual text generation process, as discussed in the next section.

In our systems, the discourse history is a simple stack. Explanation plans are large, complex structures and they will accumulate rapidly as the dialogue progresses. Exhaustively searching the discourse history for relevant prior explanations is computationally prohibitive. Therefore, we require indexing strategies that allow the system to find possibly relevant prior explanations in an efficient manner. We have found that two factors influence the identification of relevant prior explanations:

- what was conveyed in prior explanations, i.e., the relationship of the domain content to be conveyed to the domain content presented in prior explanation(s)
- why a previous explanation was given, i.e., the relationship between the communicative goal(s) of the current and prior explanation(s)

Therefore, to facilitate the search for relevant prior explanations, we index each segment of an explanation by the communicative goal(s) the text is intended to achieve, and the domain entities and situations that are discussed in that segment.[2]

Let us consider how these indices can be used to facilitate the identification of a relevant prior explanation for generating the tutor's response in turn 6 of Figure 1. When the student asks why testing pin 28 is not a good step (turn 5), the system forms the communicative goal: `(BEL H (SUBOPTIMAL-STEP (MEAS-TEST (PIN28 OF A1A3A15))))`. One of strategies for answering this question looks to see if there is a previous student action, ?A, that is similar to the test of pin 28 and that was involved in a previous explanation segment satisfying the goal `(BEL H (SUBOPTIMAL-STEP ?A))`. Using an algorithm described below, the system determines that in this case, the action of testing pin 38 is similar to the test of pin 28 and there is an explanation (turn 3) satisfying the communicative goal `(BEL H (SUBOPTIMAL-STEP`

[2]Following Grosz and Sidner [8], in our model intentions are the basic determiner of segmentation, and therefore each communicative goal indicates a segment boundary.

USER	Describe Inderal.	[1]
SYSTEM	Inderal is a drug that is used for prophylactic treatment of migraine. Drugs for prophylactic treatment are intended to reduce or prevent further migraine attacks. These are drugs that you must take every day, whether or not you have a migraine. In general, prophylactic treatment is suitable for patients with frequent migraines. Inderal may not be suitable for you if you have diabetes, problems with your heart, or a severe pulmonary disease such as asthma. Some side effects of Inderal are lowered blood pressure, fatigue, and depression.	[2]
USER	Describe Elavil.	[3]
SYSTEM	*Like Inderal,* Elavil is used for prophylactic treatment of migraine. ϕ Elavil may not be suitable for you if you have irregular heartbeat or if you experience dizziness when you stand up. ***Elavil is better than Inderal for patients who experience depression, because Elavil is actually an anti-depressant.*** However, Elavil has other possible side effects including dry mouth, difficulty urinating, and weight gain.	[4]

Figure 2: Actual Dialogue with Patient Education System

(MEAS-TEST (PIN38 OF A1A3A15))))). The explanation strategy points out the similarity between the prior and current explanation, generating the text "As explained before, . . ." in turn 6.

Other strategies cover cases in which an identical communicative goal was attempted before, or the action itself or a similar action was discussed but in service of a different communicative goal. These strategies use the two types of indices to quickly determine if there are prior explanations that satisfy the constraints on their applicability. I provide examples from the patient education domain in the Section .

Determining Relationships between Domain Entities

In the patient education system, domain knowledge is represented in LOOM [9], a term-subsumption language. Therefore, domain entities and relationships between them are well defined and determined simply by queries written in LOOM's query language.

In the Sherlock system, much of the knowledge used in troubleshooting and assessing student's actions is represented procedurally, and therefore other techniques for computing relationships between domain entities are needed. In RFU interactions, the most commonly asked question is to justify the tutor's assessment of a step (32% of all questions asked during RFU), and 27% of the answers to such questions involve references to previously assessed actions. Therefore, an efficient algorithm for computing similarity of student actions was considered essential for producing the types of context-sensitive explanations that are required in this domain. To compute similarity between actions, the system uses a technique adapted from Ashley's work in case-based legal reasoning [10]. This algorithm, developed by James Rosenblum, makes use of a set of facets that SHERLOCK employs to evaluate each student action. These facets were derived from a cognitive task analysis aimed at identifying the factors that expert avionics tutors use in assessing student's troubleshooting actions [11].

Associated with each facet is an indication of whether that facet contributes to a good (+), bad (−), or neutral (n) evaluation in the current problem-solving context. The system's representation of a student action includes the list of facets characterizing the action.

Treating each student action as a "case", the algorithm builds a *similarity DAG* representing a partial ordering of actions based on the similarity of each action to a given action. The system can compute overall similarity, or similarity with respect to a certain class of facets (+, −, or n). For example, when answering a question about why the current action received a negative assessment, the similarity DAG is built so that it indicates similarity of previous actions to the current action with respect to the − facets. The root of the DAG represents the current action and the facets that apply to it. Each node in the graph represents a set of actions that share the same set of facets. The more facets that a node has in common with the current action (the root node), the closer it will be to the root node.

Initial results using this algorithm are quite promising. The algorithm is both efficient (complexity $O(n^2)$ where n is the number of student actions) and accurate. In a corpus of 8 student-tutor protocols involving 154 student actions and 30 requests to justify the tutor's assessment of the student's action, the human tutor produced 8 responses that explicitly pointed out similarity(ies) to action(s) whose assessment had previously been explained. These 8 responses involved 11 similar actions in total. In all 8 situations the algorithm correctly selected as most similar the same actions used in the tutor's explanations. In 3 cases the algorithm suggested a similarity not used by the tutor. However, when presented with these similarities, our expert tutor judged them as correct and stated that explanations that explicitly pointed out these similarities would have been pedagogically useful. In all other cases in which the human tutor did not make reference to a previous explanation as part of an answer, our algorithm reported that no prior action was similar.

```
NAME: Op1
EFFECT: (KNOW-ABOUT H ?d))
CONSTRAINTS: (AND (ISA ?d DRUG)
                  (USE ?d ?t))
NUCLEUS: (BEL H (USE ?d ?t))
SATELLITES:
   (((BEL H (SOMEREF (contraindication ?d))) *required*)
    ((BEL H (SOMEREF (other-use ?d))) *optional*)
    ((BEL H (SOMEREF (side-effect ?d))) *required*)
    ((BEL H (SOMEREF (warning ?d))) *optional*))

NAME: Op3
EFFECT: (BEL H (?r ?arg1 ?arg2))
CONSTRAINTS: (IN-DH (BEL H (?r ?x ?arg2)))
NUCLEUS: (BEL H (SAME-AS (?r ?x ?arg2)
                         (?r ?arg1 ?arg2)))
SATELLITES: nil

NAME: Op6
EFFECT: (BEL H ?p)
CONSTRAINTS: nil
NUCLEUS: (INFORM H ?p)
SATELLITES: nil
```

Figure 3: Sample Plan Operators from Patient Education System

5. Exploiting Prior Explanations

With the capability to identify relevant prior discourse, our systems are able to exploit this information when planning explanations using three mechanisms: plan operators that implement context-sensitive strategies, domain-independent planning heuristics (e.g., prefer operators that refer to previous explanations), and plan modification rules that alter a plan based on information from the discourse history (e.g., if an optional communicative goal has already been achieved, don't plan text to achieve it).

We now consider how the patient education system can produce the behavior illustrated in the sample dialogue in Figure 2. When the user asks the system to 'Describe Inderal' (turn 1), the system posts the goal (KNOW-ABOUT H INDERAL). The planner searches its operator library to find an operator capable of achieving this goal, and finds Op1 shown in Figure 3. This operator encodes a strategy for describing a drug derived from our analysis of transcripts of doctor-patient interactions and interviews with physicians.

To determine whether this operator can be used in the current situation, the planner checks its constraints. If a constraint predicate includes only bound variables, then the planner verifies the constraint against the knowledge base. For example, the first constraint in Op1 (ISA ?d DRUG) checks the domain knowledge to verify that INDERAL is of type DRUG. Alternatively, if a constraint predicate contains variables that are not yet bound, the planner searches the system's knowledge bases for acceptable bindings for such variables. For example, to check the constraint (USE ?d ?t) where ?d is bound to INDERAL, but ?t is not bound, the planner searches the medical knowledge base and finds that the variable ?t can be bound to PROPHYLACTIC-MIGRAINE-TREATMENT. Therefore, all the constraints on Op1 are verified, and the operator is chosen. To expand the operator, the planner posts the subgoal appearing in the nucleus[3] field of the operator, (BEL H (USE INDERAL PROPHYLACTIC-MIGRAINE-TREATMENT)), and then the subgoals appearing in the satellite. Expanding the satellites of Op1 posts up to four additional subgoals.

The planner must then find operators for achieving each of the subgoals. To achieve the first subgoal, (BEL H (USE INDERAL PROPHYLACTIC-MIGRAINE-TREATMENT)), the planner uses Op6 which encodes the simple strategy: to make the hearer believe any proposition ?p, simply inform her of ?p. Speech acts, e.g., INFORM and RECOMMEND, are the primitives of our text planning system. When a subgoal has been refined to a speech act, the system constructs a *functional description (FD)* for the speech act. When text planning is complete, these FDs are passed to the FUF sentence generator [13] which produces the actual English text.

In the process of building an FD, new text planning goals may be posted as side effects. This occurs because it is only when building FDs that the planner considers how concepts will be realized in text. To provide informative and understandable explanations, the system uses the plan modification heuristic: "Post optional subgoals to explain unfamiliar terms introduced in explanation". During the process of building FDs, this heuristic causes the system to check its user model to see if each term that will be mentioned in the text is known to the user. In transforming (INFORM H (USE INDERAL PROPHYLACTIC-MIGRAINE-TREATMENT)), the interface notes that the user does not already know the concept PROPHYLACTIC-MIGRAINE-TREATMENT, therefore it posts a subgoal to describe this term, as shown in the system's utterance in turn 2 of the sample dialogue.

The rest of the explanation in turn 2 results from expanding the remaining satellite subgoals in a similar manner. The user then asks the system to describe Elavil (turn 3). Op1 is again chosen, however, this time the planner finds two applicable operators for achieving the subgoal (INFORM H (USE ELAVIL PROPHYLACTIC-MIGRAINE-TREATMENT)), namely Op3 and Op6. Note that the constraint of Op3 (IN-DH (BEL H (?r ?x ?arg2))) (where ?r is bound to USE and ?arg2 to PROPHYLACTIC-MIGRAINE-TREATMENT) is satisfied by binding ?x to INDERAL because the system

[3]The terms ***nucleus*** and ***satellite*** come from Rhetorical Structure Theory (RST). For more details about RST, see [12].

achieved the goal (BEL H (USE INDERAL PROPHYLACTIC-MIGRAINE-TREATMENT))) in its previous explanation. The system can determine this efficiently using the indices described in the previous section.

The system has a selection heuristic that guides it to prefer operators that refer to previous explanations, and thus Op3 is chosen to achieve the current goal. Refining this operator leads the system to generate the text "Like Inderal, Elavil is used for ...". Another context-sensitive operator applies when the system expands the subgoal (BEL H (OTHER-USE ELAVIL DEPRESSION)), and leads to the text "Elavil is better than Inderal ...". In addition, note that the system did not explain the term PROPHYLACTIC-MIGRAINE-TREATMENT when describing Elavil. This is because when the system attempts to determine whether this term is known to the user, it finds that the term was explained in the previous text (i.e., the goal (KNOW-ABOUT H PROPHYLACTIC-MIGRAINE-TREATMENT) appears in a previous text plan), and therefore it does not re-explain this term.

Thus we see that, by checking for the existence of certain communicative goals in the discourse history, context-sensitive plan operators, plan selection heuristics, and plan modification rules enable the system to generate context-sensitive responses.

6. Related Work

Computational linguists have investigated how the context provided by the previous discourse should affect the generation of referring expressions, including pronominalization decisions (e.g., [14, 15]). Others have studied how a more extensive discourse history could affect other aspects of the response. Swartout's XPLAIN system can suggest simple analogies with previous explanations and omit portions of a causal chain that have been presented in an earlier explanation. However, this is the only type of contextual effect implemented in XPLAIN, and it was done so using an ad hoc technique to provide this one effect. We are attempting to provide a more general approach.

McKeown carried out a preliminary analysis of how previous discourse might affect a system's response to users' requests to describe an object or compare two objects [16]. She found that by simply maintaining a list of the questions that had been asked, it was possible to avoid certain types of repetition. She further found that if the system were to keep track of the exact information that was provided previously, it could create a text that contrasts or parallels an earlier one. While McKeown's analysis was fairly detailed, no discourse history was maintained in the implementation, and none of the suggestions for how responses could be altered, if such a history existed, were actually implemented or tested. We are devising a way for explanation strategies to make use of the information stored in a discourse history, and are implementing these strategies.

Finally, our work bears some resemblance to work in *plan adaptation* [17]. Systems using plan adaptation often use CBR techniques to index a library of previously synthesized plans. However, plan adaptation is concerned with indexing plans so that they can be retrieved and reused, perhaps with modification, in later situations. Our emphasis is not on reusing plans, but rather on exploiting prior plans as one of many knowledge sources affecting explanation generation.

References

1. J. A. Rosenblum and J. D. Moore. A field guide to contextual effects in instructional dialogues. Technical report, University of Pittsburgh, Computer Science Department, forthcoming.
2. J. D. Moore and C. L. Paris. Planning text for advisory dialogues. In *Proc. of the 27th Annual Meeting of the ACL*, pp. 203–211, 1989.
3. J. D. Moore. *A Reactive Approach to Explanation in Expert and Advice-Giving Systems*. PhD thesis, University of California, Los Angeles, 1989.
4. B. G. Buchanan, J.D. Moore, D. E. Forsythe, G. E. Banks, and S. Ohlsson. Involving patients in health care: Using medical informatics for explanation in the clinical setting. In *Proc. of the Symposium on Computer Applications in Medical Care*, pp. 510–514. McGraw-Hill Inc., 1992.
5. G. Carenini and J. D. Moore. Generating explanations in context. *Proceedings of the International Workshop on Intelligent User Interfaces*, pp. 175–182, 1993. ACM Press.
6. A. Lesgold, S. Lajoie, M. Bunzo, and G. Eggan. Sherlock: A coached practice environment for an electronics troubleshooting job. In *Computer Assisted Instruction and Intelligent Tutoring Systems: Shared Goals and Complementary Approaches*, pp. 201–238. LEA Hillsdale, New Jersey, 1992.
7. J. D. Moore and W. R. Swartout. Pointing: A way toward explanation dialogue. In *Proc. of AAAI-90*, pp. 457–464, 1990.
8. B. J. Grosz and C. L. Sidner. Attention, intention, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.
9. R. MacGregor and M. H. Burstein. Using a description classifier to enhance knowledge representation. *IEEE Expert*, 6(3):41–4, 1991.
10. K. D. Ashley. *Modeling Legal Argument: Reasoning with Cases and Hypotheticals*. MIT Press, Cambridge, MA, 1990.
11. R. Pokorny and S. Gott. The evaluation of a real-world instructional system: Using technical experts as raters. Technical report, Armstrong Laboratories, Brooks Air Force Base, 1990.
12. W. C. Mann and S. A. Thompson. Rhetorical Structure Theory: Towards a functional theory of text organization. *TEXT*, 8(3):243–281, 1988.
13. M. Elhadad. FUF: the universal unifier user manual version 5.0, October 1991.
14. R. Granville. Controlling lexical substitution in computer text generation. In *Proc. of COLING*, pp. 381–384, 1984.
15. R. Dale. Cooking up referring expressions. In *Proc. of the 27th Annual Meeting of the ACL*, pp. 68–75, 1989.
16. K. R. McKeown. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge University Press, Cambridge, England, 1985.
17. R. Alterman. Adaptive planning. In Stuart Shapiro, editor, *The Encyclopedia of Artificial Intelligence*, pp. 5–15. Wiley, New York, 1992.