Generating Natural Language Descriptions with Examples: Differences between Introductory and Advanced Texts

Vibhu O. Mittal and Cécile L. Paris

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292

Department of Computer Science
University of Southern California
Los Angeles, CA 90089

Abstract

Examples form an integral and very important part of many descriptions, especially in contexts such as tutoring and documentation generation. The ability to tailor a description for a particular situation is particularly important when different situations can result in widely varying descriptions. This paper considers the generation of descriptions with examples for two different situations: introductory texts and advanced, reference manual style texts. Previous studies have focused on any the examples or the language component of the explanation in isolation. However, there is a strong interaction between the examples and the accompanying description and it is therefore important to study how both these components are affected by changes in the situation.

In this paper, we characterize examples in the context of their description along three orthogonal axes: the information content, the knowledge type of the example and the text-type in which the explanation is being generated. While variations along either of the three axes can result in different descriptions, this paper addresses variation along the text-type axis. We illustrate our discussion with a description of a `list` from our domain of LISP documentation, and present a trace of the system as it generates these descriptions.

Introduction

Examples are an integral part of many descriptions, especially in contexts such as tutoring and documentation generation. Indeed, the importance of using illustrative examples in communicating effectively has long been recognized, e.g., (Greenwald, 1984; Doheny-Farina, 1988; Norman, 1988). People like examples because examples tend to put abstract, theoretical information into concrete terms they can understand. In fact, one study found that 76% of users looking at system documentation initially ignored the description and went straight to the examples (LeFevre and Dixon, 1986). A system that generates descriptions must thus be able to include examples. Furthermore, the ability to tailor a description for a particular situation is particularly important as different situations can result in widely varying descriptions, where both the textual descriptions and the accompanying examples vary. Some researchers have already looked at how a textual description can be affected by different situations (or different users), e.g. (Paris, 1988; Bateman and Paris, 1989). Others have studied how to construct or retrieve appropriate examples, e.g. (Rissland and Soloway, 1980; Ashley and Aleven, 1992; Rissland, 1983; Suthers and Rissland, 1988). However, the issue of tailoring descriptions that include examples for the situation at hand has not been addressed. Yet, it is clear that one cannot plan a description tailored to a user, and then independently and as an afterthought, add some examples to the description: Sweller and his colleagues found that if the examples and the descriptive component were not well integrated, the combination could result in reduced user comprehension (Chandler and Sweller, 1991; Ward and Sweller, 1990). Examples and text must be presented to the user as a coherent whole, and *together*, appropriately tailored to the situation.

Because examples are crucial in documentation (Charney *et al.*, 1988; Feldman and Klausmeier, 1974; Klausmeier and Feldman, 1975; Reder *et al.*, 1986), and documentation is a critical factor in user acceptance of a system, we chose automatic documentation as our domain to investigate the issue of generating descriptions that include examples. This domain has additional advantages: there is a large body of work on documentation writing, a lot of actual material that we can study, including numerous examples of the text types we are concerned with (introductory and advanced). In previous work, we have described the issues that must be addressed for a system to be able to generate descriptions with well integrated examples (Mittal and Paris, 1992). In this paper, we show how two specific situations, introductory texts and advanced texts, result in two different such descriptions.

This paper is structured as follows: Section 2 briefly reviews the issues that arise when generating text with examples. Section 3 presents a categorization of example types that allows us to provide a characterization of the differences between the texts in introductory *vs* references manuals and Section 4 discusses these differences. Section 5 describes our text planning framework, and Section 6 presents a trace of the algorithm.

The authors gratefully acknowledge support from NASA-Ames grant NCC 2-520 and DARPA contract DABT63-91-C-0025. Cécile Paris also acknowledges support from NSF grant IRI-9003087.

A list always begins with a left parenthesis. Then come zero or more pieces of data (called the elements of a list) and a right parenthesis. Some examples of lists are:

```
(AARDVARK)
(RED YELLOW GREEN BLUE)
(2 3 5 11 19)
(3 FRENCH FRIES)
```

A list may contain other lists as elements. Given the three lists:

```
(BLUE SKY) (GREEN GRASS) (BROWN EARTH)
```

we can make a list by combining them all with a parentheses.

```
((BLUE SKY) (GREEN GRASS) (BROWN EARTH))
```

Figure 1: A description of `list` in an introductory text from (Touretzky, 1984), p.35

A `list` is recursively defined to be either the empty list or a CONS whose CDR component is a `list`. The CAR components of the CONSes are called the elements of the list. For each element of the list, there is a CONS. The empty list has no elements at all.

A `list` is annotated by writing the elements of the list in order, separated by blank space (space, tab, or return character) and surrounded by parentheses. For example:

```
(a b c)           ; A list of 3 symbols
(2.0s0 (a 1) #\*) ; A list of 3 things:a
                  ; floating point number,
                  ; another list, and a
                  ; character object
```

Figure 2: A description of `list` from a reference manual from (Steele Jr., 1984), p.26

Section 7 concludes with a look at the limitations.

Integrating Examples in Descriptive Texts

Many issues need to be considered when generating descriptions that integrate descriptive text and examples, because both these components co-constrain and affect each other. The inclusion of examples in an explanation can sometimes cause additional text to be generated; at other times, it can cause certain portions of the original explanation to be elided. A generation system must therefore take into account the interaction between the descriptive text and the examples, as well as effects from other factors, such as the presentation order of the examples, the placement of the examples with respect to each other, as well as the descriptive text, etc.

While we have discussed these issues elsewhere (Mittal and Paris, 1992; Mittal, 1993 forthcoming), we review some of them here:

- What should be in the text, in the examples, in both?
- What is a suitable example? How much information should a single example attempt to convey? Should there be more than one example?
- If multiple examples are to be presented, what is the order of presentation?
- If an example is to be given, should the example be presented immediately, or after the whole description is presented? This will determine whether the example(s) appear *within*, *before*, or *after* the descriptive text.
- Should prompts[1] be generated along with the examples?

Answers to these questions will depend on whether the text is an introductory or advanced text. Consider, for example, the descriptions of `list` given in Fig. 1 taken from (Touretzky, 1984), an introductory manual, and Fig. 2 taken from (Steele Jr., 1984), a reference manual: they contain very different information in both their descriptive portions as well as their examples; while Fig. 1 contains 8 lists (which are used either as examples or as background to the examples), Fig. 2 has only 2 examples. Finally, the examples in Fig. 1 do not contain prompts, while those in Fig. 2 do.

[1] 'Prompts' are attention focusing devices such as arrows, marks, or even additional text associated with examples (Engelmann and Carnine, 1982).

Categorizing Example Types in Context

In order to provide appropriately tailored examples, we must first characterize the type of examples that can appear in descriptions. This will then help the system in choosing appropriate examples to present as part of a description.

While some example categorizations (Michener, 1978; Polya, 1973) have already been proposed, we found these inadequate as they do not take the context of the whole explanation into account. This is because previous attempts at categorizing example types were done in an *analytical* rather than a *generational* context, and, as a result, these categorizations suffered from two drawbacks from the standpoint of a computational generation system: (i) they do not explicitly take into account the context in which the example occurred, and (ii) they did not differentiate among different dimensions of variation.

An example of how important the context is in determining the category of the example can be seen if we look at the two descriptions of a `list`, shown in Fig. 3, taken from our LISP domain. The empty list `NIL` is an anomalous example for the first definition, while it is a positive example for the second one. Thus it is clear that categorization depends upon not only the example, but the surrounding context (which includes the descriptive text accompanying the example) as well.

Based on our analysis of a number of instructional texts, numerous reference manuals and large amounts of system documentation, we formulated a three dimensional system to categorize examples by explicitly taking the context into account. The three dimensions are:[2]

1. the polarity of the example with respect to the description: It can be: (i) positive, i.e., the example is an instance of the description, (ii) negative, i.e., the example is *not* an instance of the description, or (iii) anomalous, i.e., the example either looks positive and is actually negative, or vice-versa.

[2] Further details on this classification of examples into a three dimensional space may be found in (Mittal and Paris, 1993).

A left parenthesis followed by zero or more S-expressions followed by a right parenthesis is a list.

From (Shapiro, 1986)

A `list` is recursively defined to be either the empty list or a `CONS` whose `CDR` component is a `list`. The `CAR` components of the `CONS`es are called the elements of the list. For each element of the list, there is a `CONS`. The empty list has no elements at all. The empty list `NIL` therefore can be written ( ), because it is a list with no elements.

From (Steele Jr., 1984)

Figure 3: Two definitions that cause `NIL` to be classified differently as an illustrative example.

2. the knowledge type being communicated: for example, a concept, a relation or a process is being described.

3. the genre or text-type to be generated: For now, we only take into consideration two text-types:[3] (i) descriptions in introductory texts, and (ii) descriptions in reference manuals. These are, in our case, closely related to the user types: introductory texts are intended for beginners and naive users while advanced texts are intended for expert users.[4]

Note that each of these axes can be further sub-divided (for instance, concepts can be further specified as being single-featured or multi-featured concepts, etc.).

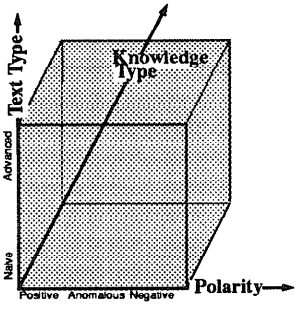

Figure 4: Three dimensions for example categorization.

Such a categorization is essential to narrow the search space for suitable examples during generation. Furthermore, it allows us to make use of the numerous results in educational psychology and cognitive science, on how to best choose and present examples for a particular text- and knowledge-type. For example, results there suggest constraints that can be taken into consideration with respect to the number of examples to present, e.g., (Markle and Tiemann, 1969), their order of presentation, e.g., (Carnine, 1980; Engelmann and Carnine, 1982), whether anomalous examples should be presented, e.g., (Engelmann and Carnine, 1982), etc.

[3]We make use of the notion of a text-type here only in a very broad sense to define distinct categories that affect the generation of examples in our framework for the automatic documentation task. However, these text-types can be refined further. Indeed, several detailed text typologies have been proposed by linguists e.g., (Biber, 1989; de Beaugrande, 1980).

[4]We have in fact referred to this axis as 'user type' in other work.

Introductory *vs* Advanced Texts

We now consider how descriptions that contain examples differ, when we move along the text-type axis of our categorization, from introductory to advanced text. We address each of the questions presented in Section 2:

The descriptive component: In the case of the introductory text-type, the descriptive component contains surface or syntactic information; in the case of the reference text-type, the description must include complete information, including the internal structure of the concept.

The actual examples: Examples in both text-types illustrate critical features[5] of the surface or syntactic form of the concept or its realization. In introductory texts, however, examples are simple and tend to illustrate only one feature at a time. (Sometimes it is not possible to isolate one feature, and an example might illustrate two features; in this case, the system will need to generate additional text to mention this fact.) On the other hand, examples in reference texts are multi-featured.

The number of examples: Since introductory texts contain usually single-featured examples, the number of examples depend upon the number of critical features that the concept possesses. In contrast, reference texts contain examples that contain three or four features per example (Clark, 1971), and, therefore, proportionately fewer examples need to be presented.

The polarity of the examples: Introductory texts make use of both positive and negative examples, but not anomalous examples. Advanced texts on the other hand, contain positive and anomalous examples.

The position of the examples: In introductory texts, the examples are presented immediately after the point they illustrate is mentioned. This results in descriptions in which the examples are interspersed in the text. On the other hand, examples in reference texts must be presented only after the description of the concept is complete.

Prompts: The system needs to generate prompts for examples that contain more than one feature. The system must also generate prompts in the case of recursive examples (they use other instances which are also instances of the concept), and anomalous examples if background text has not yet been generated (as is done for introductory texts).

These guidelines are summarized in Fig. 5.

In the following section, we will illustrate how a system can use these guidelines to generate descriptions (text and examples) for both introductory and advanced texts, in our domain of the programming language LISP.

The Generation System

Our system is part of the documentation facility we are building for the Explainable Expert Systems (EES) framework (Swartout *et al.*, 1992). The framework implements the integration of text and examples within a text-generation system. More specifically, we use a text-planning system that constructs text by explicitly reasoning about the communicative

[5]Critical features are features that are *necessary* for an example to be considered a positive example of a concept. Changes to a critical feature cause a positive example to become a negative example.

goal to be achieved, as well as how goals relate to each other rhetorically to form a coherent text (Moore and Paris, 1989; Moore, 1989; Moore and Paris, 1992). Given a top level communicative goal (such as `(KNOW-ABOUT HEARER (CONCEPT LIST))`,[6] the system finds plans capable of achieving this goal. Plans typically post further sub-goals to be satisfied. These are expanded, and planning continues until primitive speech acts are achieved. The result of the planning process is a discourse tree, where the nodes represent goals at various levels of abstraction, with the root being the initial goal, and the leaves representing primitive realization statements, such as `(INFORM ...)` statements. The discourse tree also includes *coherence relations* (Mann and Thompson, 1987), which indicate how the various portions of text resulting from the discourse tree will be related rhetorically. This tree is then passed to a grammar interface which converts it into a set of inputs suitable for input to a grammar.

Plan operators can be seen as small schemas which describe how to achieve a goal; they are designed by studying natural language texts and transcripts. They include conditions for their applicability, which can refer to the system knowledge base, the user model, or the context (the text plan tree under construction and the dialogue history). In this framework, the generation of examples is accomplished by explicitly posting the goal of providing an example while constructing the text.

A Trace of the system

We now describe a trace of the system as it plans the presentation of descriptions similar to the ones presented in Fig. 1 and 2.

First, assume we want to produce a description of a `list` for an introductory manual. The system is given a top-level goal: `(KNOW-ABOUT HEARER (CONCEPT LIST))`. The text planner searches for applicable plan operators in its plan-library, and it picks one based on the applicable constraints such as the text-type (introductory), the knowledge type (concept), etc.[7] The text-type restricts the choice of the features to present to be syntactic ones. The main features of `list` are retrieved, and two subgoals are posted: one to list the critical features (the left parenthesis, the data elements and the right parenthesis), and another to elaborate upon them.

At this point, the discourse tree has only two nodes apart from the initial node of `(KNOW-ABOUT H (CONCEPT LIST))`: namely (*i*) `(BEL H (MAIN-FEATURES LIST (LT-PAREN DATA-ELMT RT-PAREN)))`, and (*ii*) `(ELABORATION FEATURES)`,[8] which will result in a goal to describe each of the features in turn.

The planner searches for appropriate operators to satisfy these goals. The plan operator to describe a list of features indicates that the features should be mentioned in a sequence. Three goals are appropriately posted at this point. These

For each issue, the effect of the text-type is:

- Examples:
  introductory: simple, single critical-feature
  advanced: complex, multiple critical-features
- Accompanying Description:
  introductory: surface, syntactic information
  advanced: complete information, including internal structure
- Number of Examples:
  introductory: depends upon number of critical features
  advanced: few (each example contains three to four features)
- Positioning the Examples:
  introductory: immediately after points being illustrated
  advanced: after the description is complete
- Prompts:
  introductory: prompt if example has more than one feature
  advanced: prompts if anomalous and recursive examples

Figure 5: Brief description of differences between examples in introductory and advanced texts.

goals result in the planner generating a plan for the first two sentences of Fig. 1. The other sub-goal (the `ELABORATION`) also causes three goals to be posted for describing each of the critical features. Since two of these are for elaborating upon the parentheses, they are not expanded because no further information is available. So only the goal of describing the data elements remains. A partial representation of the resulting text plan is shown in Fig. 6.[9]

Data elements can be of three types: numbers, symbols, or lists. The system can either communicate this information by realizing an appropriate sentence, or through examples – since it can generate examples for each of these types, or both. The text type (introductory text) constraints cause the system to pick examples. (If the text-type had been 'reference,' the system would have delayed the presentation of examples, and text would have been generated at that point instead of the examples.) The system posts two goals to illustrate the two dimensions along which the data elements can vary: the number of elements and the type.

Information about a particular feature can be communicated by the system through examples efficiently by using pairs (or groups) of examples as follows:

- if the feature to be communicated happens to be a *critical* feature, the system generates pairs of examples, *one positive and one negative*, which are identical except for the feature being communicated, and
- if the feature to be communicated happens to be a *variable*[10]

[6] See the references given above for details on the notation used to represent these goals.

[7] When several plans are available, the system chooses one using *selection heuristics* designed by (Moore, 1989).

[8] `ELABORATION` is one of the coherence relations defined in (Mann and Thompson, 1987).

[9] All the text plans shown in this paper are simplified versions of the actual plans generated: in particular, the communicative goals are not written in their formal notation, in terms of the hearer's mental states, for readability's sake.

[10] Variable features are features that can *vary* in a positive example. Changes to variable features creates different positive examples.

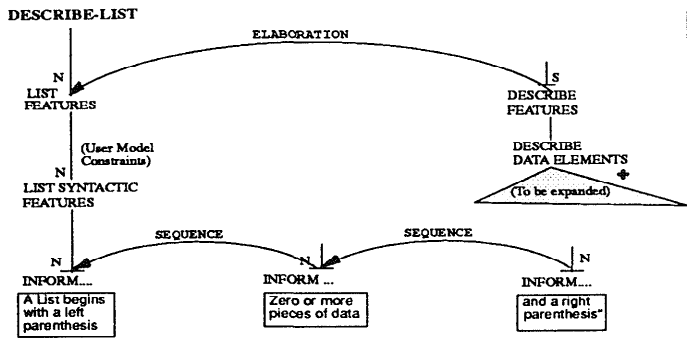

Figure 6: Skeletal plan for listing main features of `list`.

feature, the system generates pairs of examples that are *both positive* and are widely different in that feature

Thus, to communicate the fact that there can be any number of elements, the system posts two goals to generate two differing positive examples, one with a single element and another with multiple elements. The example generation algorithm ensures that the examples selected for related sub-goals (such as the two above) differ in *only* the dimension being highlighted. However, as the examples contain two critical features (i.e., type is illustrated as well), the system generates prompts to focus the attention on the reader on the number feature ("a list of one element" *vs* "a list of several elements").

The goal to illustrate the *type* dimension is handled in similar fashion, with four sub-goals (one each for the types: symbols, numbers, symbols and numbers, and sub-lists) being posted. The last data type, *sub-lists*, is marked by the algorithm as a *recursive* use of the concept, and is handled specially because the text-type is introductory. In the case of an introductory text, such examples must be introduced with appropriate explanations added to the text. (If the text-type had been 'reference,' the system would have generated a prompt denoting the presence of the sub-list.) The resulting skeletal text-plan generated by the system is shown in Fig. 7.

Consider the second case now, when the text-type is specified as being 'reference.' In this case, the system starts with the same top-level goal as before, but the text-type constraints cause the planner to select both the structural representation of a `list`, as well as the syntactic structure for presentation. The system posts two goals, one to describe the underlying structure, and one to describe the syntactic form of a `list`. The two goals expand into the first two paragraphs in Fig. 2. Note that the examples occur at the end of the description. The two examples generated are much more complex than the previous case, and they contain a number of variable features (the second example shows the variation in element types, as well as the variation in number possible). Since the second example generated contains a list as a data element, the system generates prompts for the examples. For lack of space, the resulting text plan is not presented here.[11]

In both of the above cases, the completed discourse tree is passed to an interface which converts the `INFORM` goals into

[11] See (Mittal, 1993 forthcoming) for more details.

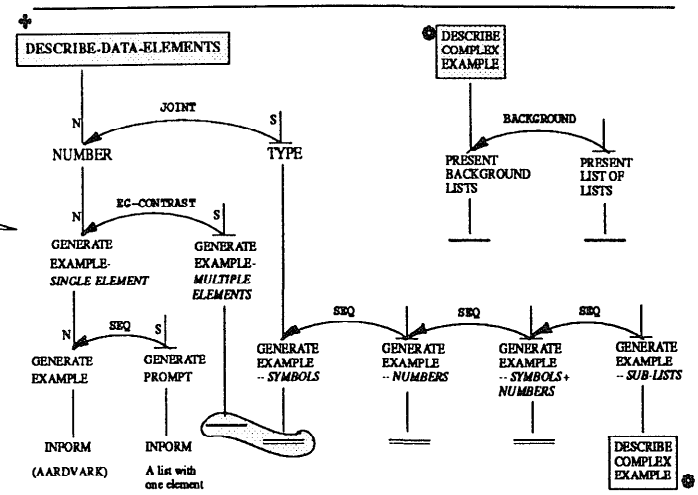

Figure 7: Skeletal plan for generating examples of `lists`.

the appropriate input for the sentence generator. The interface chooses the appropriate lexical and syntactic constructs to form the individual sentences and connects them appropriately, using the rhetorical information from the discourse tree.

Conclusions

We have presented an analysis of the differences in descriptions that integrate examples for introductory and advanced texts. To be able to do this, we first presented a brief description of our characterization of examples, explicitly taking into account the surrounding context. Variation along any of these axes causes the explanation generated to change accordingly. This variation occurs not just in the descriptive part of the explanation, but also in the examples that accompany it. Since the examples and the descriptive component are tightly integrated and affect each other in many ways, a system designed to generate such descriptions must take into account these interactions and be able to structure the presentation accordingly. We have presented information necessary to generate descriptions for two text-types: introductory and advanced. The algorithm used by the system was illustrated by tracing the generation of two descriptions of the LISP `list`.

The issues we have described are not specific to a particular framework or implementation for generation of either text or examples. In fact, the algorithm described is implemented in our system as constraint specification across different plan operators. We have successfully combined two well-known generators (one for text and one for examples) in our system to produce the explanations described in this paper.

References

Ashley, K. D. and Aleven, V., 1992. Generating Dialectical Examples Automatically. In *Proceedings of AAAI-92*, 654–660. San Jose, CA.

Bateman, J. A. and Paris, C. L., 1989. Phrasing a text in terms the user can understand. In *Proceedings of IJCAI 89*. Detroit, MI.

Biber, D., 1989. A typology of English Texts. *Linguistics* 27:3–43.

Carnine, D.W., 1980. Two Letter Discrimination Sequences: High-Confusion-Alternatives first versus Low-Confusion-Alternatives first. *Journal of Reading Behaviour*, XII(1):41–47, Spring.

Chandler, P. and Sweller, J., 1991. Cognitive Load Theory and the Format of Instruction. *Cognition and Instruction* 8(4):292–332.

Charney, D. H., Reder, L. M., and Wells, G. W., 1988. Studies of Elaboration in Instructional Texts. In Doheny-Farina, S.(Ed.), *Effective Documentation: What we have learned from Research*, 48–72. Cambridge, MA. The MIT Press.

Clark, D. C., 1971. Teaching Concepts in the Classroom: A Set of Prescriptions derived from Experimental Research. *Journal of Educational Psychology Monograph* 62:253–278.

de Beaugrande, R., 1980. *Text, Discourse and Process*. Ablex Publishing Co.

Doheny-Farina, S., 1988. *Effective Documentation : What we have learned from Research*. MIT Press.

Engelmann, S. and Carnine, D., 1982. *Theory of Instruction: Principles and Applications*. New York: Irvington Publishers, Inc.

Feldman, K. V. and Klausmeier, H. J., 1974. The effects of two kinds of definitions on the concept attainment of fourth- and eighth-grade students. *Journal of Educational Research* 67(5):219–223.

Greenwald, J., 1984. How does this #%$! Thing Work? *Time*. (Page 64, Week of June 18, 1984).

Klausmeier, H. J. and Feldman, K. V., 1975. Effects of a Definition and a Varying Number of Examples and Non-Examples on Concept Attainment. *Journal of Educational Psychology* 67(2):174–178.

Klausmeier, H. J., 1976. Instructional Design and the Teaching of Concepts. In Levin, J. R.*et al.*(Eds.), *Cognitive Learning in Children*. New York: Academic Press.

LeFevre, J.-A. and Dixon, P., 1986. Do Written Instructions Need Examples? *Cognition and Instruction* 3(1):1–30.

Mann, W. and Thompson, S., 1987. Rhetorical Structure Theory: a Theory of Text Organization. In Polanyi, L. (Ed.), *The Structure of Discourse*. Norwood, New Jersey: Ablex Publishing Co..

Markle, S.M. and Tiemann, P.W., 1969. *Really Understanding Concepts*. Stipes Press, Urbana, IL.

Michener, E. R., 1978. Understanding Understanding Mathematics. *Cognitive Science Journal* 2(4):361–383.

Mittal, V.O. and Paris, C.L., 1992. Generating Object Descriptions which integrate both Text and Examples. In *Proc.9th Canadian A.I. Conference*, pp. 1–8. Morgan Kaufmann Publishers.

Mittal, V. O. and Paris, C. L., 1993. Categorizing Example Types in Context: Applications for the Generation of Tutorial Descriptions. To appear in the *Proceedings of AI-ED 93*. (Edinburgh, Scotland).

Mittal, V.O., 1993 (forthcoming). *Generating descriptions with integrated text and examples*. PhD thesis, University of Southern California, Los Angeles, CA.

Moore, J. D. and Paris, C. L., 1989. Planning text for advisory dialogues. In *Proceedings of ACL 89*. Vancouver, B.C.

Moore, J.D. and Paris, C.L., 1992. Planning text for advisory dialogues: Capturing intentional, rhetorical and attentional information. TR 92-22, Univ. of Pittsburgh, CS Dept., Pittsburgh, PA, 1992.

Moore, J. D., 1989. A Reactive Approach to Explanation in Expert and Advice-Giving Systems. Ph.D. thesis, UCLA, CA.

Norman, D., 1988 *The Psychology of Everyday Things*. New York: Basic Books.

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

Polya, G., 1973. *Induction and Analogy in Mathematics*, volume 1 of *Mathematics and Plausible Reasoning*. Princeton, N.J.: Princeton University Press.

Reder, L. M., Charney, D. H., and Morgan, K. I., 1986. The Role of Elaborations in learning a skill from an Instructional Text. *Memory and Cognition* 14(1):64–78.

Rissland, E. L. and Soloway, E. M., 1980. Overview of an Example Generation System. In *Proc. AAAI 80*, pp. 256–258.

Rissland, E. L., 1980. Example Generation. In *Proceedings of the 3rd Conference of the Candian Society for Computational Studies of Intelligence*, 280–288. Toronto, Ontario.

Rissland, E. L., 1983. Examples in Legal Reasoning: Legal Hypotheticals. In *Proc. IJCAI 83*, pp. 90–93. Karlsruhe, Germany.

Shapiro, S.C., 1986. *LISP: An Interactive Approach*. Rockville, MD.: Computer Science Press.

Steele Jr., G. L., 1984. *Common Lisp: The Language*. Digital Press.

Suthers, D. D. and Rissland, E. L., 1988. Constraint Manipulation for Example Generation. COINS TR 88-71, CIS Dept, Univ. of Massachusetts, Amherst, MA.

Swartout, W. R., Paris, C. L., and Moore, J. D., 1992. Design for Explainable Expert Systems. *IEEE Expert* 6(3):58–64.

Touretzky, D. S., 1984. *LISP: A Gentle Introduction to Symbolic Computation*. New York: Harper & Row Publishers.

Ward, M. and Sweller, J., 1990. Structuring Effective Worked Examples. *Cognition and Instruction* 7(1):1 – 39.