AD-A278 685

(12)

University of Southern California

Categorizing Example Types in Instructional Texts: The Need to consider Context

Vibhu O. Mittal and Cecile Paris
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

February 1993
ISI/RR-93- 332

DTIC
S ELECTE D
APR 2 8 1994
G

128 17

[illegible]

DTIC QUALITY INSPECTED 3

INFORMATION SCIENCES INSTITUTE

310/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

Best
Available
Copy

12

Categorizing Example Types in Instructional Texts: The Need to consider Context

Vibhu O. Mittal and Cecile Paris
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

February 1993
ISI/RR-93- 332

Accesion For		
NTIS CRA&I	✓	
DTIC TAB	☐	
Unannounced	☐	
Justification		
By		
Distribution /		
Availability Codes		
Dist	Avail and / or Special	
A-1		

DTIC
ELECTE
APR 2 8 1994
S G D

94-12817

DTIC QUALITY INSPECTED 3

[illegible]
Approved for public release
[illegible]

94 4 26 114

REPORT DOCUMENTATION PAGE	*FORM APPROVED* *OMB NO. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching exiting data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimated or any other aspect of this collection of information, including suggestings for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE February 1993	3. REPORT TYPE AND DATES COVERED Research Report

4. TITLE AND SUBTITLE Categorizing Example Types in Instructional Texts: The Need to consider Context	5. FUNDING NUMBERS NCC2-250 DABT63-91-C-0025
6. AUTHOR(S) Vibhu O. Mittal and Cecile L. Paris	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) USC INFORMATION SCIENCES INSTITUTE 4676 ADMIRALTY WAY MARINA DEL REY, CA 90292-6695	8. PERFORMING ORGANIZATON REPORT NUMBER RR-332
9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES) ARPA 3701 Fairfax Street Alexandria, VA 22203; NASA Ames Moffett Field, CA 94035	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Accepted to AI-ED '93 (World Conference on AI in Education, Edinburgh, Scotland, 1993 AACE)

12A. DISTRIBUTION/AVAILABILITY STATEMENT UNCLASSIFIED/UNLIMITED	12B. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Different situations often require the presentation of different types of examples with specific presentation requirements about the number of examples, sequences of presentation, the associated prompts, etc. A specification of the different presentation requirements is particularly important in designing an efffective ITS. A categorization of example types and their associated presentation requirements is necessary. In this paper, we argue that examples must be characterized based on the context in which they appear, and present one such characterization, and describe how these can be effectively used by an ITS to generate tutorial descriptions that incorporate examples.

14. SUBJECT TERMS examples, instructional texts, context, intelligent tutoring systems	15. NUMBER OF PAGES 8
	16. PRICE CODE

17. SECURITY CLASSIFICTION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNLIMITED

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reoprts. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay within the lines to meet optical scanning requirements.

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month,a nd year, if available (e.g. 1 jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element numbers(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the repor.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ...; To be published in... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17.-19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contins classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

Accepted to AI-ED '93
(World Conf. on AI in Education

Categorizing Example Types in Instructional Texts: The Need to consider Context

Vibhu O. Mittal and Cécile L. Paris

Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
U.S.A.

Department of Computer Science
University of Southern California
Los Angeles, CA 90089-0782
U.S.A.

Abstract: Different situations often require the presentation of different types of examples with specific presentation requirements about the number of examples, the sequence of presentation, the associated prompts, etc. A specification of the different presentation requirements is particularly important in designing an effective ITS. A categorization of example types and their associated presentation requirements is necessary. In this paper, we argue that examples must be characterized based on the context in which they appear, and present one such characterization, and describe how these can be effectively used by an ITS to generate tutorial descriptions that incorporate examples.

Introduction

Examples play an important role in comprehension, e.g., (Houtz, Moore, and Davis 1973; Pirolli 1991; Reder, Charney, and Morgan 1986), and it is therefore important for an intelligent tutoring system to be able to present examples to the learner. A large number of examples can potentially be used to illustrate a given point. However, not all examples are equally effective in all situations; some are better than others in specific contexts, and others tend to illustrate different aspects of the same concept in different ways and achieve different goals. Categorizing examples is useful because identifying a category from which to generate an example can greatly constrain the number of possible examples that can be applicable in the given situation.

Previous studies on the categorization of examples include studies by Polya (1945), and Michener (1978), on the suitability of examples in different situations. However, these categorizations did not explicitly take into account the *context* in which the example was presented. Yet, the context of an example affects its characterization and usefulness. To use examples effectively – i.e., as an important and a complementary part of the overall description – the system must reason with the constraints introduced by both the textual explanation, as well as the examples (Mittal and Paris 1992; 1993). This is because both the *examples and the surrounding description affect each other.*

There are many issues that must be considered in selecting and presenting examples, e.g., (Bruner 1966; Engelmann and Carnine 1982). In this paper, we address the issue of characterizing the *type of examples* that appear in tutorial descriptions, as this can help a system in choosing appropriate examples to present. In the following sections, we describe previous work on categorizing example types, and illustrate how the same example can be categorized in two different categories if the accompanying description is not taken into account. Finally, we present our categorization, taking into consideration the context, based on three orthogonal dimensions: the *information-content*, the *intended-audience* and the *knowledge-type* of the example.

Previous Work on Categorizing Examples

Polya (1973) categorized examples into three categories: (*i*) *leading examples*, (*ii*) *suggestive examples*, and (*iii*) *counter examples*. Leading examples were ones that contained mostly *critical*[1] features and very few "irrelevant features;" they were meant for naive users. Suggestive examples contained more *variable*[2] features than leading examples and were meant to "guide the student in the correct direction." Counter-examples were negative examples that illustrated how instances were *not* indicative of some concept.

In her work, Michener categorized examples into five categories (Michener 1977; 1978): (*i*) *introductory examples:* perspicuous, simple cases, (*ii*) *model examples:* general, paradigmatic cases, (*iii*) *reference examples:* standard, ubiquitous cases, (*iv*) *counter examples:* limiting, falsifying cases, and (*v*) *anomalous examples:* exceptional, pathological cases.

We believe that both categorizations suffer from two problems:

1. because they do not explicitly take into account the context of the presentation, the same example can often be classified into different categories
2. the definition of the category is not clearly specified; it is therefore difficult to implement in a computational system

Furthermore, the two categorizations above did not specify relationships (if any) between their different categories, nor did they specify whether these categories were mutually exclusive.

The Necessity of Categorizing Examples based on the Context

Our categorization of examples was driven by the need to be able to generate tutorial and explanatory descriptions that integrate examples coherently in a computational framework.[3] In such a framework, the system must be able to present suitable examples to illustrate the description or the definition being presented. The suitability of an example is usually determined in the context it appears in, rather than in the abstract: it depends upon the goal of the description, what features are being presented, where in the overall description the example appears, etc.

Furthermore, the suitability of the example is also affected by other examples around it. A number of studies on the cognitive effectiveness of examples have shown that the presentation order of the examples plays an important role in user comprehension, e.g., (Carnine 1980; Litchfield, Driscoll, and Dempsey 1990). Thus, the appropriateness of one example, presented for the same description, can be different, based on other examples that appear with it, and where it appears. It is therefore obvious that an example can be categorized only in conjunction with the context in which it appears.

We shall now describe the three dimensions along which we characterize an example in context: the relationship of the information in the example to that in the context, the intended audience of the example, and the knowledge type being communicated by the examples.

The First Dimension: The relationship between the example and the description

One of the dimensions that an example can be characterized along is the relationship of the information contained in the example with the information contained in the accompanying descriptive explanation that it illustrates. Along this dimension, an example can fall into three categories:

(AARDVARK)	; example of a list
AARDVARK	; not a list

Figure 1: Two examples about a `list`.

Positive Examples: These examples are instances of the concept being described and satisfy the properties of the concept as described in the accompanying description. These examples must possess all the critical features of the concept they illustrate. Such examples are usually in an *elaborative* role to the information in the description.

Negative Examples: Negative examples (or counter-examples) are *not* instances of the concept being described.

[1]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.

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

[3]Further details on this work on the design and implementation of a natural language system capable of integrating examples and text can be seen in (Mittal and Paris 1992; 1993; Mittal 1992).

These are cases that do *not* meet the requirements specified in the accompanying description, and they play a *contrastive* role in the context. Negative examples can be very useful, because they help rule out non-critical features of a concept (Houtz, Moore, and Davis 1973). For instance, the examples of a `list` in the programming language LISP in Figure 1 illustrate the need for parentheses in a `list`. The negative example conveys the information that the symbol AARDVARK by itself is not sufficient for an instance to be a list. By virtue of the fact that the only difference between a positive and a negative example is the set of parentheses, it draws attention to the fact that the parentheses are important for something to be a `list`. Thus, ***features in common*** between positive and negative examples ***can be ruled out as sufficient*** features, while ***differing features*** are highlighted as ***necessary*** features and thus become more important.

Anomalous Examples: Anomalous examples represent irregular or exceptional cases. These are either: (*i*) instances of the concept described, but not covered by the description, or (*ii*) those are likely to be mis-classified by the user (because of an incomplete description). Thus, positive instances which appear to be very different from other positive examples, or negative instances which appear to be very similar to positive examples, would be classified as anomalous cases. Anomalous examples must be presented with appropriate introductory text, and presented apart from the other examples (Engelmann and Carnine 1982).

The classification of an example into either of these categories depends upon the context established by the accompanying descriptive explanation. As mentioned previously, it is possible that an example which would be classified as an anomalous example in one context could classified as a normal, positive example in another context. Consider the following description of a `list` in LISP:

> A left parenthesis followed by zero or more S-expressions followed by a right parenthesis is a list.
>
> From (Shapiro 1986)

Given the above definition of a `list`, the following examples would classify as positive, negative and anomalous cases:

Positive Examples	*Negative Examples*	*Anomalous Examples*
(A B C D)	'THIS-IS-AN-ATOM	NIL
(1 2 3 4 5 6 7)	1234567	(a . b)
(BLUE SKIES GREEN GRASS)	'BLUE	

This categorization of examples could change with another definition:

> A `list` is a CONS-cell whose CDR is either the atom `NIL` or another list. The atom `NIL` is the identifier that represents the empty list and the boolean concept `FALSE`.
>
> From (Steele Jr. 1984)

In this case, `NIL` ***becomes a positive example*** of a `list`. Similarly, a `list` may be so defined as to include the concept of a dotted-list as well.

It is clear that it is difficult, and sometimes impossible, to classify an example as belonging to a certain category without taking into consideration the surrounding contextual information. It is also difficult to categorize examples as being 'suggestive' or 'model' or 'reference' without having a complete definition of these different categories. It is also not possible to label an example 'positive' or 'negative' without knowing the definition it is supposed to illustrate (AARDVARK is positive example of an `atom`, but is a negative example of a `list`). In addition, an example that is 'anomalous' in one context can classify as a positive example in another context. Correct classification of the examples is essential, because examples must be presented in accordance with the category they happen to classify in. For instance, anomalous examples can cause great confusion in an introductory user if presented along with other positive examples. (Anomalous examples should be presented separately from the regular examples, with a suitable introduction to notify the user of the anomalous nature of such examples.)

The Second Dimension: The intended audience

The second dimension that examples can be characterized along is dictated by the intended audience type of the presentation. This is an important constraint on the selection of information to be presented *both* in the description

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))
```

From (Touretzky 1984), page 35.

Figure 2: Introductory examples are usually single featured.

A list looks like a sequence of objects, without commas between them, enclosed in parentheses.

⋮

Appropriately constructed lists can also be used to call functions in LISP. If you type any of the lists in table 2-4 to LISP, you will get an appropriate response.

```
Table 2-2:
  (1 2 3 4 5)         ; List of numbers
  (A B C D)           ; List of symbols
  (#\A #\B #\C #\D)   ; List of characters
Table 2-3:
  (This is (also) a list)
  ("this is a string in a list" -53)
  ((Beth "555-5834") (Pat "555-8098"))
Table 2-4:
  (SQRT 2)
  (+ 2 3)
  (- 6 5 4)
```

Lists can be considered ways to store data. For example, you might want to store your inventory as a list, or group together names and phone numbers in a list.

From (Tatar 1987), page 16.

Figure 3: Intermediate 'use' oriented examples.

and the example. There have been many studies on the need for varying both the amount of information and the manner of its presentation, based on the user, e.g., (Paris 1988; Nwana 1991; London 1992). These studies have demonstrated that there are significant differences in descriptions and examples meant for different user types.

As we have already mentioned before, the major short-coming of both the previous example categorizations was due to the fact that they did not take the accompanying context into account. In contrast, we consider *both* the description and the example for categorization. This is essential in our case, because the system needs to generate both the text as well as the example in its explanation. Often, even though the examples tend to look alike, the accompanying descriptions are very different for different user types. For instance, Pirolli found that in some domains, such as recursion, the examples presented to both naive and advanced users were almost identical, but their explanations were very different (Pirolli 1991; Pirolli and Anderson 1985). Feldman and Klausmeier found similar differences in the phrasing of definitions presented to fourth and eight grade students (Feldman and Klausmeier 1974).

From our analyses of naturally occurring texts, we have classified examples (in the context of their accompanying descriptions) into three main classes – *introductory, intermediate* and *advanced*. This classification constrains the content and the presentation style of the descriptions and the examples used with them:

1. *introductory:* – users with little or no previous exposure assumed for the concept; goal is to *learn about* the concept,
2. *intermediate:* – users with moderate previous exposure; goal is to *learn to make use* of the concept,
3. *advanced:* – users with extensive knowledge; goal is to clarify some point or misconception about the concept.

Introductory Users: Examples in introductory descriptions[4] tend to be simple ones – where 'simple' refers to the fact that they are usually single-featured (or if they have multiple features, sometimes two, where the two features are along two different feature dimensions). This has also been reported in other studies, e.g., (Clark 1971; Michener 1977; Carnine 1980; Litchfield, Driscoll, and Dempsey 1990). In our domain of LISP descriptions, the accompanying description is syntactic or surface/appearance oriented. Anomalous examples are usually absent,

[4]We shall use terms such as 'introductory descriptions' to indicate descriptions meant for an introductory audience.

and if they are presented, they are done so *after* all the other examples. Examples are often introduced as soon as the point they illustrate is mentioned in the text.

Consider for instance the description in Figure 2. The descriptions are centered around the syntax or the surface appearance of the `list`. The examples are simple and illustrate a feature at a time (the *type* of data elements, except in one case where the *type* and the *number*, two different dimensions of variation, are illustrated). Examples do not always have prompts,[5] because the same information is often realized as sentences in the accompanying description.

Intermediate Users: Descriptions written for the 'intermediate' user (who is already assumed to have introductory knowledge) tend to be more complex than the ones for introductory users, in that they include more detail on *how* the information may be *used* by the user. The examples are not always presented immediately; if there are a number of related points, these points are stated first, before a group of examples illustrating these points are presented. The examples themselves are usually briefly annotated (with prompts). Intermediate descriptions contain a few introductory examples, which are then followed by typical uses of such example instances, which contain mostly multi-featured examples. For example, the description in Figure 3 describes how a `list` can be used to represent shopping lists, store phone numbers and write function calls.

Advanced Users: Since the purpose of advanced or reference materials is *not* instruction, it is not surprising that both the textual description and the accompanying examples are very different from those in the introductory ones. The documentation and the examples usually occur in a fixed format, with the examples following the definition and the explanation. The examples are not simple, single-featured, but tend to be few and multi-featured (typically three to four features). The examples are often almost independent of the textual description, with little cross-referencing between the two. This almost invariably results in prompts being used to indicate some of the salient characteristics of the examples. Since the descriptions tend to be comprehensive, there are few (if any) anomalous examples. If there are any anomalous examples, they are *always* presented. For example, a description of a `list` from an advanced, reference manual is shown in Figure 4.

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
                     ; float. point number,
                     ; another list, and a
                     ; character object
```

The empty list `NIL` therefore can be written ( ), because it is a list with no elements.

From Steele (Steele Jr. 1984), p.26

Figure 4: Reference documentation has few examples.

The Third Dimension: The Knowledge-Type

In addition to the *user-type* and the *example-type* which can be used to constrain the possible choices that need to be made in generation, the *knowledge-type* can also be used during the generation process to determine the appropriate type and sequence of examples to be generated in an explanation. The *knowledge-type* refers to the categorization of information into one of three broad classes: *concepts*, *relations* or *processes*. There can be significant differences in the presentation of examples and the accompanying descriptions based on whether the idea to to be explained is a concept, relation or a process.

The `list` function takes any number of inputs and makes a list of them all. For example:

INPUT to `list`		OUTPUT
'foo 'bar 'baz	→	(foo bar baz)
'foo	→	(foo)
'(frob)	→	((frob))

From (Touretzky 1984), p.51

Figure 5: Examples of a relation.

Consider for instance the *concept* 'list' (as described in Figure 2) and the *relation* 'list' (functions are relations that hold between the input parameters and the output values of the function), as described in Figure 5.

[5] 'Prompts' are additional text or markers associated with examples to draw attention to specific features in the examples (Engelmann and Carnine 1982).

The *concept* list is described as an object, and examples of list are instances which exemplify the term 'list'; the *function* list, on the other hand, is described in terms of its input and output parameters, and examples of the function reflect this fact. Similarly, *processes*, which are sequences of functions are described differently and their examples are often instances of function parameters at every step in the sequence. In generating examples of relations, it is important to keep in consideration that the examples used as input-output parameters must be known to the hearer. Also, since anomalous or pathological examples of concepts used as either input or output examples for examples of relations often result in anomalous examples of relations, the system must choose these examples carefully.

Examples of processes consist of chains of events that take place in a particular order. The goal is to communicate the *sequence of events* and their *cumulative effect*. In case the user does not know about certain relations or concepts involved in the steps of the routine, the generator must adequately explain such relations or concepts too. There are two broad ways of treating such 'gaps' in the hearer's knowledge. The missing information is given as a by-note in the textual description accompanying the example if the gap is a relatively minor one. If the knowledge required is more than just a trivial information gap, then the generator should try and postpone presenting examples using that section of the routine until the end of the sequence. This is to ensure that the hearer is familiar with the rest of the steps in the sequence before the difficult examples are encountered.

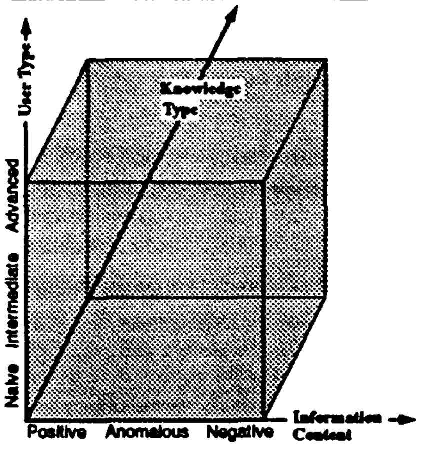

Figure 6: The three dimensions along which examples can be categorized in context.

Discussion: The dimensions along which we categorize examples are not limited to the gradations mentioned in this paper. In our framework, there are yet finer gradations which are used by the system in making decisions. For instance, *concepts* are further sub-divided into *single-featured*, *multiple-featured*, or *comparative* concepts. These finer gradations allows us to make better decisions about both the number of examples as well as their presentation order in our system. Figure 6 shows a representation of the three dimensions in this categorization.

Applications to the Generation of Tutorial Descriptions

The categorization of examples (in the context of their accompanying description) that we have outlined is extremely useful in constructing a system for generating tutorial descriptions. Our major goal is not just the selection (or generation) of appropriate examples by themselves, but the generation of a description that integrates examples and text in an effective manner. This requirement brings up many issues that may otherwise be not considered as important: issues such as the interaction between the examples and the description (how the text changes because of the presence of examples), the placement of the examples in relation to the explanation (before, within or after the description), etc.

Our current framework implements the generation of examples within a text-generation system by explicitly posting the goals of providing examples. Our system uses a planning mechanism: given a top level communicative goal, such as `(DESCRIBE LIST)`, the system finds plans capable of achieving this goal.

1. Determine if it is a "higher-order" noun (if it has named sub-types).
2. Pick positive examples that show the different sub-types. Put them together in the sequence.
3. Determine individual features that are shared by the all the positive examples that have been mentioned in Step #2.
4. Construct negative examples that have only one of the above critical features. This will prevent generalization on the basis of only that feature.
5. Put these together in a sequence based on *maximum positive divergence* and *minimum negative difference* principles.

Figure 7: Constructing Higher-Order Noun Sequences

Plans typically post further sub-goals to be satisfied, and planning continues until primitive speech acts – i.e., directly realizable in English – 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. In the discourse tree, the discourse goals are related through coherence relations. This tree is then passed to a grammar interface which converts it into a set of inputs suitable for input to a natural language generation system: PENMAN (Mann 1983).

Examples are generated by explicitly posting a goal within the text planning system: i.e., some of the plan operators used in the system include the generation of examples as one of their steps, when applicable. This ensures that the examples embody specific information that either illustrates or complements the information in the accompanying textual description. Issues such as the number of examples to be presented, the order in which they should be presented, whether they should have prompts associated with them, etc. can then be determined in conjunction (using the constraints imposed on the selection) with the categorization of the examples to be presented. Associated with each gradation in our categorization, we have specific presentation heuristics for the examples and their descriptions. Figure 7 shows one such set of directives on the presentation of examples (for a multi-dimension, higher-order concept for a naive user) represented in our system.

The resulting discourse structure is then processed to make final decisions, such as the choice of lexical items. Finally, the completed discourse tree is passed to a a system that converts the INFORM goals into an intermediate form that is accessible to Penman, which generates the desired English output.

Conclusions

The categorization of examples, along with specific guidelines of when and how different types of examples should be presented is an extremely important issue in the design of an intelligent tutoring system. Our categorization is a generalization of the previous work by Michener and Polya, and extends the scope of the characterization to take into account the surrounding context of the example. The categories along each of the three dimensions that we have mentioned are only meant to illustrate how they affect the examples and the text to be presented. These categories can be sub-divided further into smaller classes and specific presentation methods can be associated with each class.

This categorization is not specific to a particular architecture for tutoring systems, and can be easily incorporated into any system such as CEG (Suthers and Rissland 1988) or HYPO (Ashley 1991). The dimensions can be further refined or modified if necessary to suit particular applications: for instance, recent work on categorizing *dialectical examples* (Ashley and Aleven 1992) can be easily incorporated into our framework.

We have implemented a system that plans the presentation of coherent text and examples (Mittal 1992; Mittal and Paris 1992; Mittal and Paris 1993). The plan operators (which select information to be presented), also make use of the information along the *user-type* and the *knowledge-type* dimensions to structure the content as well as the surface form of the description appropriately. Thus, the characterization's modular nature allows the represented information to be shared among different resources in the system.

References

Ashley, K. and V. Aleven (1992). Generating dialectical examples automatically. In *Proceedings of AAAI-92*, San Jose, CA., pp. 654–660. AAAI.

Ashley, K. D. (1991, June). Reasoning with cases and hypotheticals in HYPO. *International Journal of Man-Machine Studies 34*(6), 753–796.

Bruner, J. S. (1966). *Toward a Theory of Instruction*. London, U.K.: Oxford University Press.

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

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.

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

Feldman, K. V. and H. J. Klausmeier (1974, January). 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.

Houtz, J. C., J. W. Moore, and J. K. Davis (1973). Effects of Different Types of Positive and Negative Examples in Learning "non-dimensioned" Concepts. *Journal of Educational Psychology 64*(2), 206–211.

Litchfield, B. C., M. P. Driscoll, and J. V. Dempsey (1990, Winter). Presentation Sequence and Example Difficulty: Their Effect on Concept and Rule Learning in Computer-Based Instruction. *Journal of Computer-Based Instruction 17*(1), 35–40.

London, R. (1992). Student modeling to support multiple instructional approaches. *User Modeling and User-Adapted Interaction 2*(1–2), 117–154.

Mann, W. C. (1983). An Overview of the Penman Text Generation System. In *Proceedings of AAAI-83*, Washington, D.C., pp. 261–265.

Michener, E. R. (1977, February). *Epistemology, Representation, Understanding and Interactive Exploration of Mathematical Theories.* Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

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

Mittal, V. O. (1992). Elaboration in object descriptions through examples. In *Proceedings of ACL-92*, Newark, Delaware, pp. 315–317.

Mittal, V. O. and C. L. Paris (1992). Generating Object Descriptions which integrate both Text and Examples. In *Proceedings of the 9th Canadian Artificial Intelligence Conference*, pp. 1–8. Morgan Kaufmann Publishers.

Mittal, V. O. and C. L. Paris (1993). Automatic Documentation Generation: The Interaction between Text and Examples. To appear in the *Proceedings of IJCAI-93*, Chambery, France.

Nwana, H. S. (1991). User modelling and user adapted interaction in an intelligent tutoring system. *User Modeling and User-Adapted Interaction 1*(1), 1–32.

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

Pirolli, P. (1991). Effects of Examples and Their Explanations in a Lesson on Recursion: A Production System Analysis. *Cognition and Instruction 8*(3), 207–259.

Pirolli, P. L. and J. R. Anderson (1985). The Role of Learning from Examples in the Acquisition of Recursive Programming Skills. *Canadian Journal of Psychology 39*, 240–272.

Polya, G. (1945). *How to Solve it – A New Aspect of Mathematical Method.* Princeton, New Jersey: Princeton University Press.

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

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 E. L. Rissland (1988). Constraint Manipulation for Example Generation. COINS Technical Report 88-71, Dept. of CIS, Univ. of Massachusetts, Amherst, MA.

Tatar, D. G. (1987). *A Programmer's Guide to COMMON LISP.* Digital Press.

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