ARI Research Note 90-45

BATBook: An Online Book and Problem Solving Environment for the Study of Skill Acquisition

Jeremiah M. Faries and Brian J. Reiser

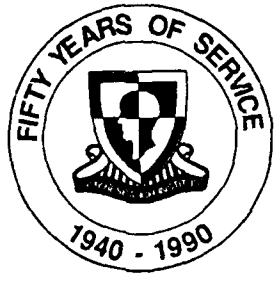
Princeton University

for

Contracting Officer's Representative
Judith Orasanu

Basic Research
Michael Kaplan, Director

July 1990



United States Army
Research Institute for the Behavioral and Social Sciences

U.S. ARMY RESEARCH INSTITUTE FOR THE BEHAVIORAL AND SOCIAL SCIENCES

A Field Operating Agency Under the Jurisdiction of the Deputy Chief of Staff for Personnel

EDGAR M. JOHNSON
Technical Director

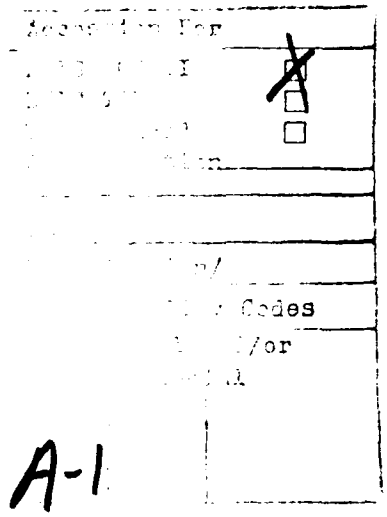
JON W. BLADES
COL, IN
Commanding

Research accomplished under contract for the Department of the Army

Princeton University

Technical review by

George Lawton



NOTICES

DISTRIBUTION: This report has been cleared for release to the Defense Technical Information Center (DTIC) to comply with regulatory requirements. It has been given no primary distribution other than to DTIC and will be available only through DTIC or the National Technical Information Service (NTIS).

FINAL DISPOSITION: This report may be destroyed when it is no longer needed. Please do not return it to the U.S. Army Research Institute for the Behavioral and Social Sciences.

NOTE: The views, opinions, and findings in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other authorized documents.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS --
2a. SECURITY CLASSIFICATION AUTHORITY --	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE --	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) --	5. MONITORING ORGANIZATION REPORT NUMBER(S) ARI Research Note 90-45
6a. NAME OF PERFORMING ORGANIZATION Princeton University 6b. OFFICE SYMBOL (If applicable) --	7a. NAME OF MONITORING ORGANIZATION U.S. Army Research Institute Office of Basic Research
6c. ADDRESS (City, State, and ZIP Code) Cognitive Science Laboratory Princeton University Princeton, NJ	7b. ADDRESS (City, State, and ZIP Code) 5001 Eisenhower Avenue Alexandria, VA 22333-5600
8a. NAME OF FUNDING / SPONSORING ORGANIZATION U.S. Army Research Institute for the Behavioral and Social Sciences 8b. OFFICE SYMBOL (If applicable) PERI-BR	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA903-87-K-0652
8c. ADDRESS (City, State, and ZIP Code) Office of Basic Research 5001 Eisenhower Avenue Alexandria, VA 22333-5600	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 61102B PROJECT NO.: 74F TASK NO.: WORK UNIT ACCESSION NO.:

11. TITLE (Include Security Classification)
BATBook: An Online Book and Problem Solving Environment for the Study of Skill Acquisition

12. PERSONAL AUTHOR(S)
Faries, Jeremiah; and Reiser, Brian

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Interim	FROM 88/05 TO 89/10	1990, July	32

16. SUPPLEMENTARY NOTATION
Contracting Officer's Representative, Judith Orasanu

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Problem solving. Memory.
Skill acquisition. Decision making
Learning. Instructional systems.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

BATBook is an online book and problem-solving environment that facilitates students' use of examples in a text book and use of their own solutions to previous problems. BATBook contains facilities for working on problems and storing solutions and for reading and searching text and examples within the text. All interaction with the system is recorded. BATBook serves as an experimental tool for studies of the use of examples in learning. BATBook has been used for studies of skill acquisition in the programming domain and can be used in similar fashion for other domains such as statistics, mathematics, or physics.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Michael Kaplan	22b. TELEPHONE (Include Area Code) (202) 274-8722 22c. OFFICE SYMBOL PERI-BR

DD Form 1473, JUN 86 *Previous editions are obsolete.*

BATBOOK: AN ONLINE BOOK AND PROBLEM SOLVING ENVIRONMENT FOR THE STUDY OF SKILL ACQUISITION

CONTENTS

BATBook: An Online Book and Problem Solving Environment for the Study of Skill Acquisition

Jeremiah M. Faries
Brian J. Reiser
Princeton University

Abstract

BATBook is an online book and problem solving environment that facilitates students' use of examples in a text book and use of their own solutions to previous problems. BATBook contains facilities for reading and searching text and examples within the text, for working on problems and storing solutions, and for searching through past solutions. All interaction with the system is recorded, and thus BATBook serves as an experimental tool for studies of the use of examples in learning. BATBook has been used for studies of skill acquisition in the programming domain, and can be used in similar fashion for other domains such as statistics, mathematics, or physics.

1 Introduction: The role of examples in learning

An important focus of research on skill acquisition is the role of examples in learning. Students rely heavily on examples in instructional text to learn procedures for solving problems in new domains (e.g., Chi, Bassok, Lewis, Reimann, & Glaser, 1989; LeFevre & Dixon, 1986; Sweller & Cooper, 1985; VanLehn, 1986; Zhu & Simon, 1987). Research on the early stages of learning has investigated how students use text examples and their own memories for previous solutions to adapt for new problems (e.g., Anderson, 1987a; Escott & McCalla, 1988; Faries & Reiser, 1988; Pirolli & Anderson, 1985; Reed, Dempster, & Ettinger, 1985; Ross, 1984; 1987; 1989). Research on analogical reasoning has considered the mechanisms that access potential analogues from memory and evaluate them for their application to a current situation or problem (e.g., Gentner, 1988; 1989; Holyoak & Thagard, in press; Thagard, Holyoak, Nelson, & Gochfeld, 1989). Recent work on case-based

reasoning has explored models of expertise consisting of an organized memory for individual cases, rather than merely consisting of generalizations (Hammond, 1989; Kolodner, 1983; Riesbeck & Schank, 1989). Research has also focused on the types of features that govern memory search for previous examples (e.g., Faries & Reiser, 1988; Gentner, 1988; Ross, 1987; 1988; Seifert, 1988; Seifert & Hammond, 1989).

Several difficult issues face research on the use of examples in learning. First, if the goal is to characterize how examples are used to learn rules, plans, or procedures, then it is important to study the use of examples during the problem solving process. Therefore it is necessary to have subjects learn to solve problems in a new domain, and to examine the learning in a period that is long enough to include a sequence of problems in which earlier solutions are relevant for later problems. It may be difficult to generalize from experiments giving subjects only material to read and remember to what happens when students are learning to solve new types of problems. Second, in many experiments it may be difficult to determine when subjects are using an example in a text. Some method is required for monitoring which pages of a text and which portions of the pages subjects are currently using. Third, even videotaping students' use of a text does not make available the information the subjects used to access previous text. Fourth, it may be difficult to distinguish a case in which a subject has retrieved a memory of a past solution and applied it to a new problem from a case in which the subject has learned a general procedure and then applied that to the target problem. In general, an optimal methodology for the study of examples would make it possible to investigate in an extended problem solving context the information subjects use to access examples and past work as well as to monitor what information from the text or past work subjects use in a new problem.

One answer to several of these problems relies on protocol analysis, in which subjects are asked to talk aloud while trying to solve a problem or understand an example in a text. Protocol analyses are one approach to gathering more fine-grained data for use in building and evaluating learning models (Newell & Simon, 1972; Ericsson & Simon; 1984). For example, Anderson and his colleagues have studied students learning to write computer programs or to solve geometry proofs and have examined 30-40 hours of learning time per subject (Anderson, 1982; Anderson, Farrell, & Sauers, 1984), and Chi and her colleagues have made a similar study of students' elaboration of examples of worked solutions while reading instructional text (Chi et al., 1989). However, it is often desirable to both restrict the range of data gathered and to find a somewhat less intrusive method for collecting the data.

In this paper, we describe the Behavioral Analogy Tracing Environment (BATBook), a computer-based environment in which students can learn to solve problems in a new domain. BATBook is an online book and problem solving environment that makes the search behavior of students explicit and provides a general method to ex-

amine how students use examples in text and memories for their previous solutions during learning.

2 BATBook: Facilitating and Monitoring the Use of Examples

BATBook is an electronic book and problem solving environment designed to facilitate and monitor students' use of text, examples, and their own previous solutions to problems. Thus, BATBook is designed to be both a pedagogical tool and an experimental tool for the study of skill acquisition.

2.1 Functionality of BATBook

BATBook can be used in skill acquisition experiments to present instructional material and monitor students' problem solving behavior. The basic paradigm is that students read a textbook on the computer screen, interspersed with example solutions to problems and with problems for the student to solve. Students progress forward and backward through the text by pages and can search the expository text and worked solutions for any particular content they can specify. While working on exercises, students are free to search the instructional material in the text, the examples contained in the text, and their past work. Students store their completed solutions and can request to see a previous solution at any time. BATBook records all interactions, including the time spent reading each page, all the successful and failed searches, all problem solving work, and searches of examples and previous solutions.

We have used BATBook to study the use of examples by students learning to write computer programs (Faries, 1988; Faries & Reiser, 1988; Faries & Reiser, in preparation). This version of BATBook contains the text of the first three chapters of *Essential LISP*, a textbook on the LISP programming language (Anderson, Corbett, & Reiser, 1987), and a problem solving environment consisting of a LISP interpreter, a simple editor, and facilities for checking the students' solutions for correctness and providing simple feedback. In addition to searching the text, the examples, and previous completed solutions, students can also search their interactions with the LISP interpreter. In this way, the BATBook environment makes explicit a large portion of the student's problem solving behavior, providing a rich record for analyzing when and how students access the written instruction and their previous work. Students learned to use the BATBook system with little trouble following a short demonstration of its capabilities, and worked with it between 10

and 15 hours to learn the three chapters worth of material, making effective use of the searching capabilities the system provides to help use the text material and their own solutions when working on new problems. The following sections describe the various components of the BATBook environment.

2.2 Implementation

The BATBook environment runs on Sun workstations in the SunView window system. BATBook is written in C, Franz LISP, GNU Emacs LISP (Stallman, 1987), and the Lex lexical analyzer (Lesk, 1975). The majority of the interface for displaying and searching text is written in C, using Lex string handling functions. The problem solving environment for LISP is written in Franz LISP, including a modified LISP interpreter that records the student's interactions and checks completed solutions. The current version of BATBook contains a total of 5800 lines of C code, 3200 lines of Franz Lisp and GNU Emacs Lisp code, and 90 lines of Lex code.

We have used several different versions of BATBook in our experiments. The exact configuration of windows and the searching options available to the students differ from one experiment to the next. In the following sections we characterize the various options available in the systems used to date, which are designed to include a LISP programming environment, but we also describe how BATBook can be used to teach topics other than programming.

2.3 BATBook Windows

The standard BATBook screen contains a Text Window, an Exercises Window, a LISP Interpreter Window, and a Problem Submission Window that appears when the student stores a completed solution (see Figure 1). The textbook is read and searched in the Text Window. The book and chapter title are displayed as the title of this window. The Text Window is replaced by a Previous Problems Window when students want to look at text examples or previous solutions (see Section 5). The Exercises Window presents problems for the student to solve. In the LISP domain, students construct their solutions and test them in the LISP Interpreter Window. For other domains, this window would be replaced by another type of interface for constructing solutions, such as an equation editing interface or a simple text editor for constructing verbal answers to problems.

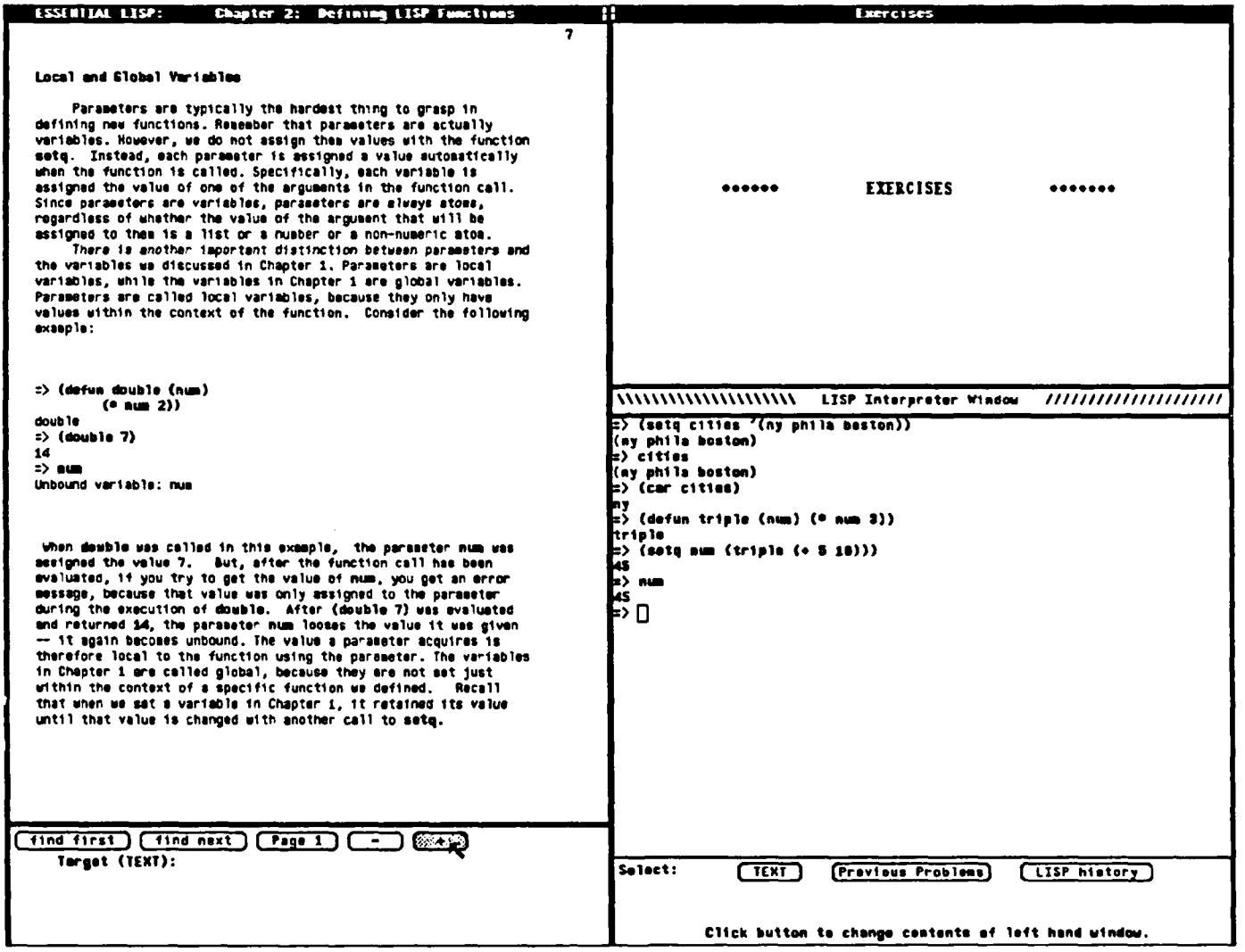


Figure 1. Reading text in the BATBook electronic book and problem solving environment.

3 Reading the Text in the Online Book

3.1 Reading and Paging

The Text Window is positioned on the left half of the screen, and displays the equivalent of approximately one page of text from a printed book. The text is displayed in the window in a 12 point font, and may contain boldface characters for marking key words, for displaying chapter headings, and for emphasis.

Students can read and page through the text using simple mouse commands. When finished reading a page, the student can select the page forward button, labeled "+," whereupon the next page of text is displayed. Students can also page backward using the button labeled "-", and can return to the first page of the chapter by selecting the "Page 1" button. The design of the Text Window was loosely based on the Superbook electronic book environment (Remde, Gomez, & Landauer, 1987).

The input text consists of a simple ASCII file and is set up like the input text for formatters such as Troff. BATBook will fill the text, insert section headings, insert paragraph and page breaks, switch between the Roman and bold font, and perform other simple formatting using Troff-like commands.

Each section ends typically with a series of examples for the student to read or exercises to solve. In most of our experiments, students are not permitted to page forward to read the next section of text until they have completed the problem set.

3.2 Searching the Text

Students frequently need to refer to previous sections of an instructional text while reading or solving problems. With BATBook, students can search through the current chapter of the text using a search capability similar to that provided in most word processing programs. The student types a target of one or more words after the "Target" prompt and then selects the "find first" or "find next" button. The "find first" button searches the text for the first occurrence of the search string. The "find next" button searches the text beginning on the current page. If the cursor is currently residing on a match, then it finds the next occurrence of the string, otherwise it simply searches for the string from the top of the current page. If the search succeeds, the page containing the target string is displayed in the Text Window, and the string is displayed in reverse video. An example of a student's search for a string in the text is shown in Figure 2.

A string of any length may be typed as the search key. The string may include spaces. Thus students may search for phrases such as "last element of a list." The

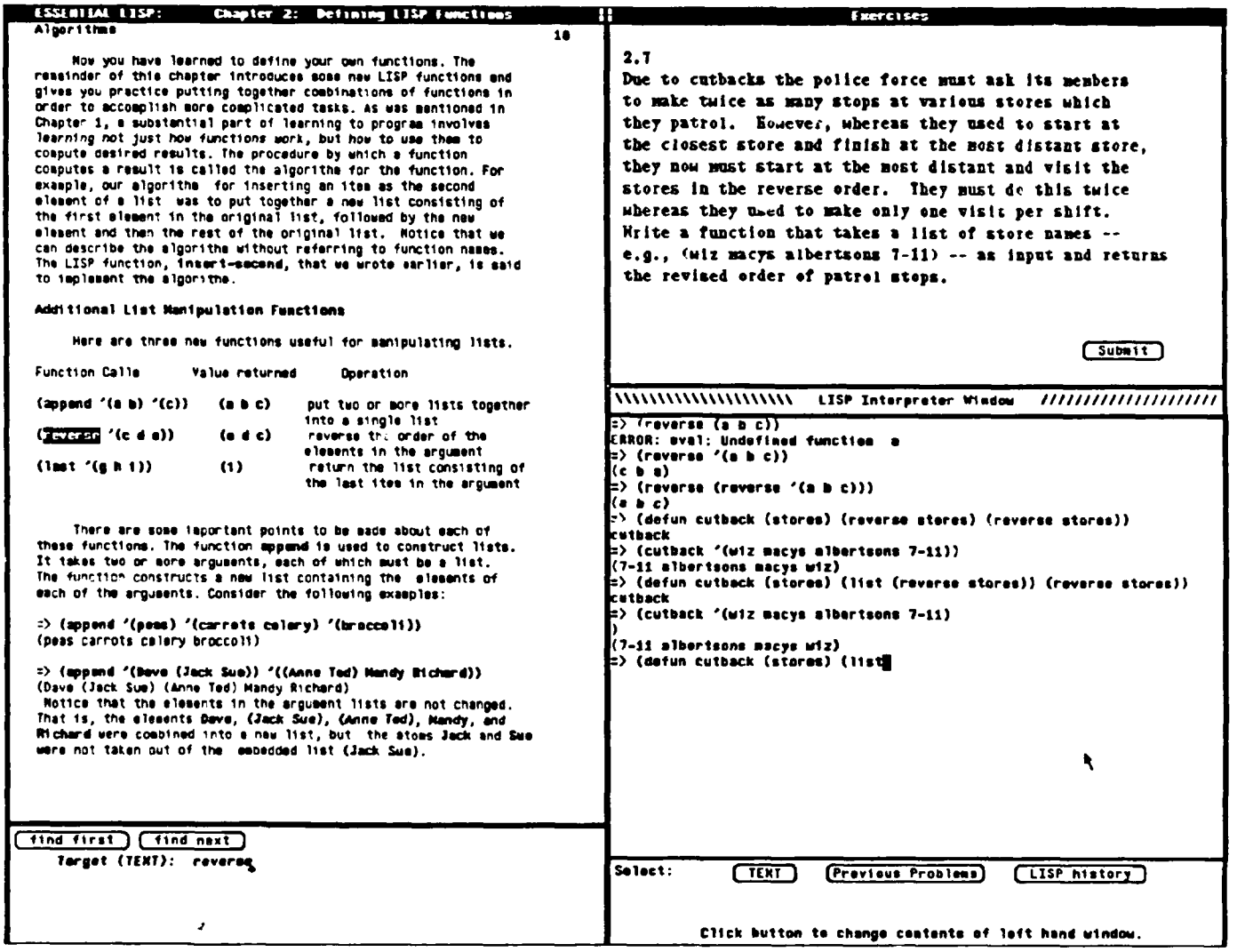


Figure 2. Searching the text and working on a problem in BATBook.

search routines are insensitive to font changes so that students can search for a phrase that was printed in boldface, such as technical terms or chapter headings. Searches are also insensitive to case changes, so searches will not fail simply because the target word began a sentence. Finally, searches are also insensitive to line breaks and page breaks, so a search for a phrase that is broken across a line or a page will succeed.

The search in BATBook differs from the search in some word processing programs in that the displayed material in BATBook is always a single page. Unlike word processing programs which display a "window" on the file, BATBook always displays pages. These page breaks are constant — if students search or page through the text, a given page will always be displayed with the same material and containing the same page number. In this way, BATBook pages are as much like physical book pages as possible, with the added advantages for the student that they can be easily paged and searched, and the advantages for the experimenter that it is possible to know at any point in the problem solving what page the subject has available for reading and exactly what target information led the subject to that particular page.

Another way that students can search previous text in a less directed fashion is to use the page forward and backward buttons to simply flip through the text until a useful section is found. In some experiments, it may be desired to restrict paging of the text while working on problems. BATBook allows the experimenter to disable paging while the student is working on a problem set. In this way, if students want to find something in the text, they must type an explicit search key instead of just paging forward or backward until they find a useful section.

Students find the ability to search text quite useful and often use it while working on exercises, particularly in the earlier portion of each chapter. This enables them to access and review text presenting new concepts required in the exercises and to find examples demonstrating the use of these concepts. Students are typically quite effective in using the search capabilities. The majority of searches are successful, and students typically follow unsuccessful searches with a successful search for a different key. In most cases students appear to recognize the context for which they were searching. A typical search pattern begins with a "find first" followed by several "find next" clicks in very quick succession, followed by the student's resumption of work on the problem using the information contained in the resulting page displayed in the Text Window.

The assumption in the design of BATBook is that the student is using the system to learn the new material contained in the instructional text rather than as a tool for browsing reference material. For this reason, search of the text is restricted to the portion of the text the student has read so far; the search will not find occurrences of the search key beyond the farthest page that has been read.

It is important to note that *any* of the content of the instructional text may be used as a key to search the text. This use of full text indexing in BATBook is similar to the search facility in the SuperBook system (Remde et al., 1987). Remde et al. argued that the ability to use any content of the text as a keyword for retrieval is an important feature of electronic books. A difficulty facing information retrieval is that users often describe targets in different terms from those built into a system. The disparity of terms naturally chosen by users of systems creates problems for information retrieval systems and for naming commands in computer systems (Furnas, Landauer, Gomez, & Dumais, 1984). Therefore, students are much more likely to be successful in retrieving a target portion of text if they can use whatever aspects of the text they remember, rather than relying on an index of keywords that the author has selected as important.

The approach of full text indexing differs from Hypertext systems currently receiving much attention (e.g., Conklin, 1987; Halasz, Moran, & Trigg, 1987; Robertson, McCracken, & Newell, 1981). Hypertext requires author-generated links between screens of information. In Hypertext, certain words or phrases in the text are marked, indicating that a related section of text is available and can be accessed if desired. Our goal in BATBook is to provide the students the freedom to pursue their own paths through the material. We do not want to draw the associations between potentially related sections through hypertext links. Instead, we are interested in seeing what paths the students themselves choose to follow and which sections or problems trigger access of previous sections. Furthermore, in our experiments students are learning to solve problems in a new topic. An important component of the target skill is knowing what information is relevant in each problem situation. Therefore, we want the responsibility for accessing relevant text to be in the hands of the student.

We feel that BATBook provides an interesting extension of the concept of an "electronic book" that has been suggested as a revolutionary new communication medium (e.g., Weyer, 1985; Yankelovich, Meyrowitz, & van Dam, 1985). It has been argued that electronic books and hypermedia provide readers access to information of various media enabling them to pursue their own paths through "webs" of connected information. BATBook enables students to access information they themselves have generated, along with what is already encoded in the book. Students using BATBook can retrieve solutions of problems along with the text when useful in solving new problems.

4 Solving Problems in BATBook

In this section, we describe how we have used BATBook to teach students to program in LISP. This version relies on a LISP problem solving environment. To use BATBook for another domain would require adapting this problem solving environment. For example, a simple interface could be added to enable students to enter equations instead of LISP expressions to work on algebra or physics problems. Completed solutions to word problems could be stored away and searched just like the LISP problems we have discussed. Even without such an interface, however, it would be possible to use the current version of BATBook in other domains by having students write their solutions on paper and use the system to read and search the text and to access examples contained in the text.

4.1 Constructing a Solution

When the student completes the reading for a section, BATBook presents the associated set of exercises. The student initiates the problem sequence by selecting the "Start Problems" button in the Exercises Window, whereupon the first problem in the set is displayed. In the LISP domain, these problems require writing LISP programs. Students have access to a LISP Interpreter Window which contains a specially written Common LISP interpreter. This interpreter prevents certain types of drastic errors (for example, accidentally redefining built-in LISP primitives or getting trapped in infinite loops) and contains a simplified error handler that avoids the standard LISP "break loop" which is often confusing for novices. The interpreter, like the other facilities in BATBook, also records every student keystroke and mouse click, as well as the resulting system response.

The environment also contains an extremely simplified Emacs-based editor which can be invoked from the LISP Interpreter Window to edit a function definition. The editing commands are invoked from labeled function keys on the keyboard. The editor contains commands to delete a character, delete a line, and insert a deleted line. The arrow keys can be used to move left and right one character and up and down one line. The editor also contains commands to save the function and enter the revised definition into LISP, and to abort the edit with the definition unaltered. The editor prevents the student from saving an illegally parenthesized expression. Students find this editor relatively easy to learn. The typical scenario for solving problems in the environment involves typing a function definition into the LISP Interpreter Window, trying the function on some examples, and then repeatedly editing the definition and trying it on examples until it works properly.

4.2 Submitting a Completed Problem

When each problem is completed, the student submits the solution in order to store it for future use. In our experiments, the submission procedure also included an analysis of the solution to determine whether it was correct. However, this is an option that can be manipulated. In some circumstances it may be desired instead to accept all student solutions without checking their correctness.

The submission process is initiated by clicking the "Submit" button that appears beneath the problem statement in the Exercises Window (see Figure 2). BATBook then prompts the student to enter the name of the function he or she has written to solve this problem (see Figure 3). BATBook then checks the solution by running the student's program on several test cases associated with each problem. If the program does not produce the correct results for a test case, BATBook informs the student how the program behaved incorrectly and asks the student to try to fix the solution. The student is required to attempt to fix the function and submit a second solution. If the second submitted solution is still incorrect, the student is again informed but this time is given the option of continuing with the next problem or continuing to try to fix the solution (see Figure 4).

After the completed solution is submitted, a new problem is displayed in the Exercises Window. At the completion of the exercises within the problem set, the Exercises Window is cleared and the Text Window displays the page that begins the next section of text.

4.3 Labeling Completed Solutions

BATBook contains an option in which students are able to label their final submitted solution (whether correct or incorrect) for the problem. In this version of the system, when students submit each solution they are asked to type in a brief label to describe it. Students are told that the label can be used at a later time to retrieve the problem description and the solution. The procedures for retrieving solutions are described in the next section.

The facility to label a completed solution enables students to retrieve their past work according to their own description of the problem and solution rather than having to refer specifically to the content of the problem or the solution itself. We included this option because we are interested in seeing whether students can profitably use the labeling facility to encode their solutions selecting features useful for later retrieval.

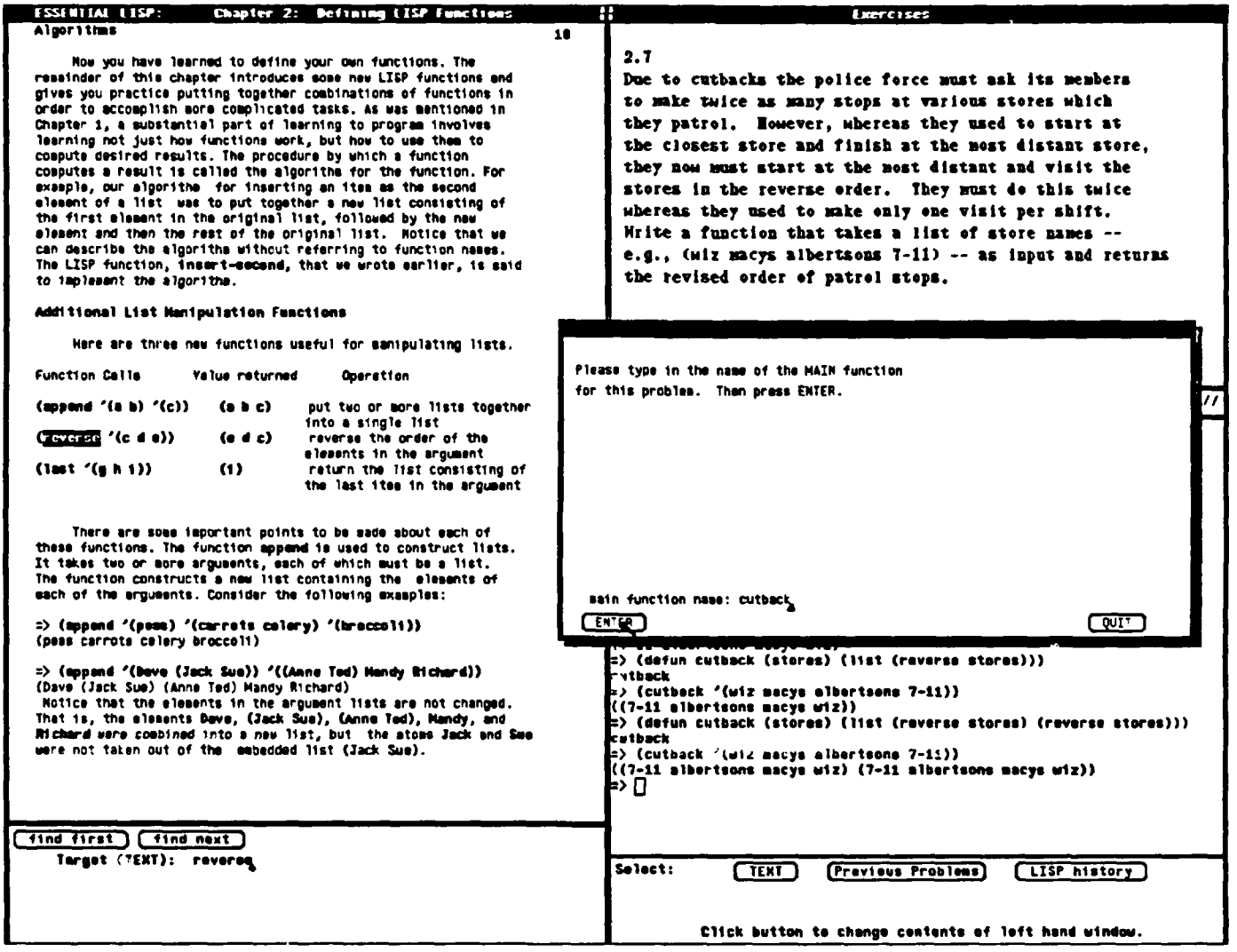


Figure 3. Submission of a completed solution.

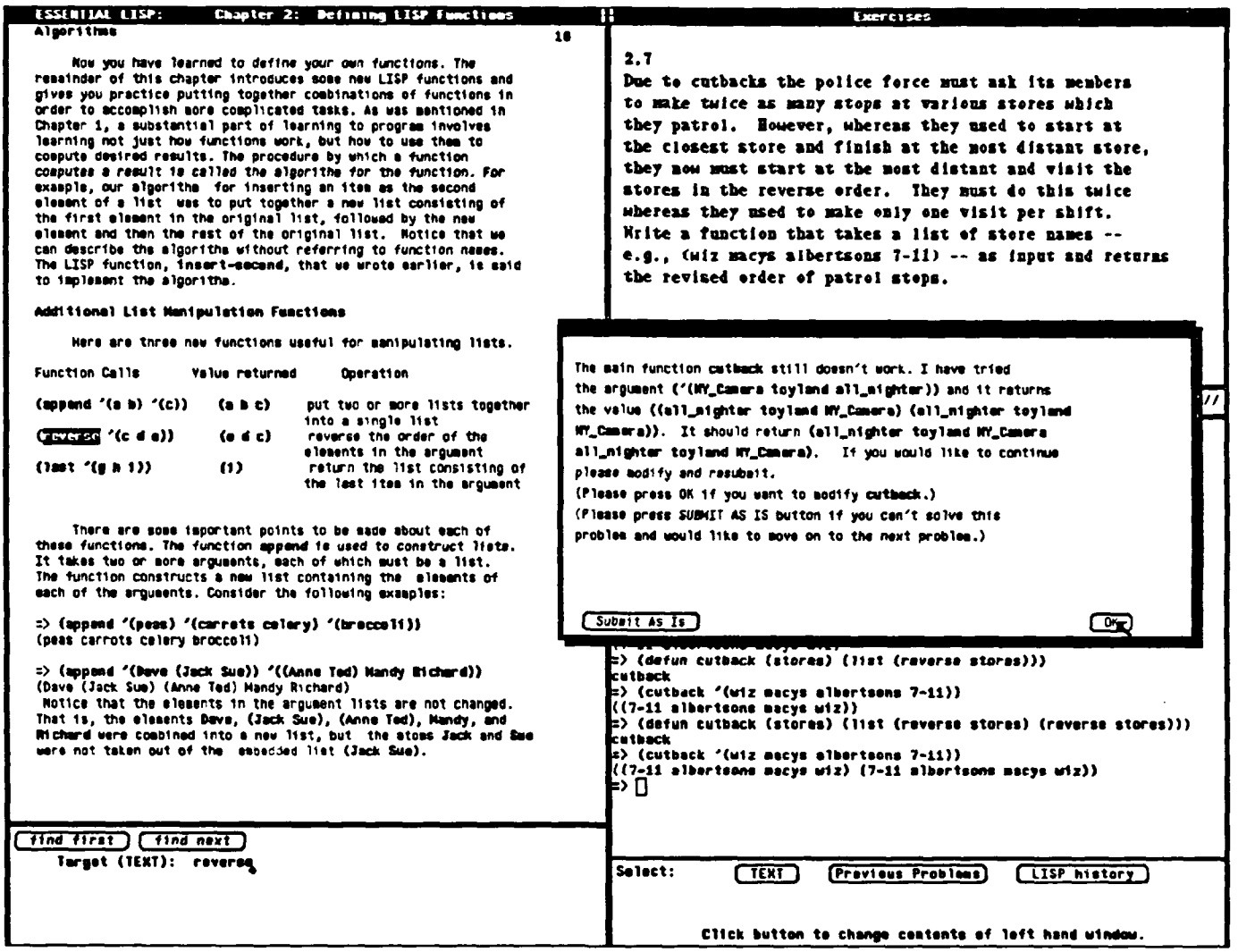


Figure 4. Feedback on a submitted incorrect solution.

5 Retrieving Past Examples

There are several methods for retrieving examples in BATBook. Two basic types of examples can be retrieved: examples within the instructional text and the student's own previous problem solving work. The examples of worked solutions embedded within the text have a special status and can be searched using a method different from the keyword text search described in Section 3.2. The method of retrieving text examples is described in Section 5.1.

The method of retrieving the student's own work depends upon the options set by the experimenter or instructor. If the student has labeled the completed solutions, these labels are used to access previous solutions. This option is described in Section 5.2. If the students were not asked for labels, they can later access their solutions by using a probe from either the problem description or the content of the solution itself. This method is described in Section 5.3.

BATBook provides another method for retrieving previous work other than completed solutions. Students can also search through their interactions with the LISP interpreter. This type of search is described in Section 5.4.

5.1 Searching Study Examples

BATBook contains the option of embedding examples within the instructional text. We refer to these as "study examples" to distinguish them from the student's own solutions to problems. A series of study examples can be included in the text similar to a series of exercises for the student to solve. Each study example includes a problem description, a solution, and an explanation of the solution (see Figure 5). The examples are presented on the right side of the screen in an expanded Exercises Window. Rather than solving the problem, the student simply reads the example and proceeds to the next example by selecting the "Next Example" button.

The student can search through any of the previous examples at any time. To initiate a search of examples, the student first selects the "Previous Problems" button beneath the LISP Interpreter Window. This changes the Text Window from Text mode to Examples mode. The examples search is most useful while solving problems, so it is important that the Exercises and LISP Interpreter Window remain in view. For that reason, the retrieved examples are placed in the Text Window. The student then types a search key opposite the "Target" prompt beneath the window, and selects the "find first" or "find next" button. The "find first" button finds the first occurrence of the search key in any of the examples that have been read so far. The "find next" button can be used to find the next occurrence, starting from the currently displayed match. If the search succeeds, then the entire example

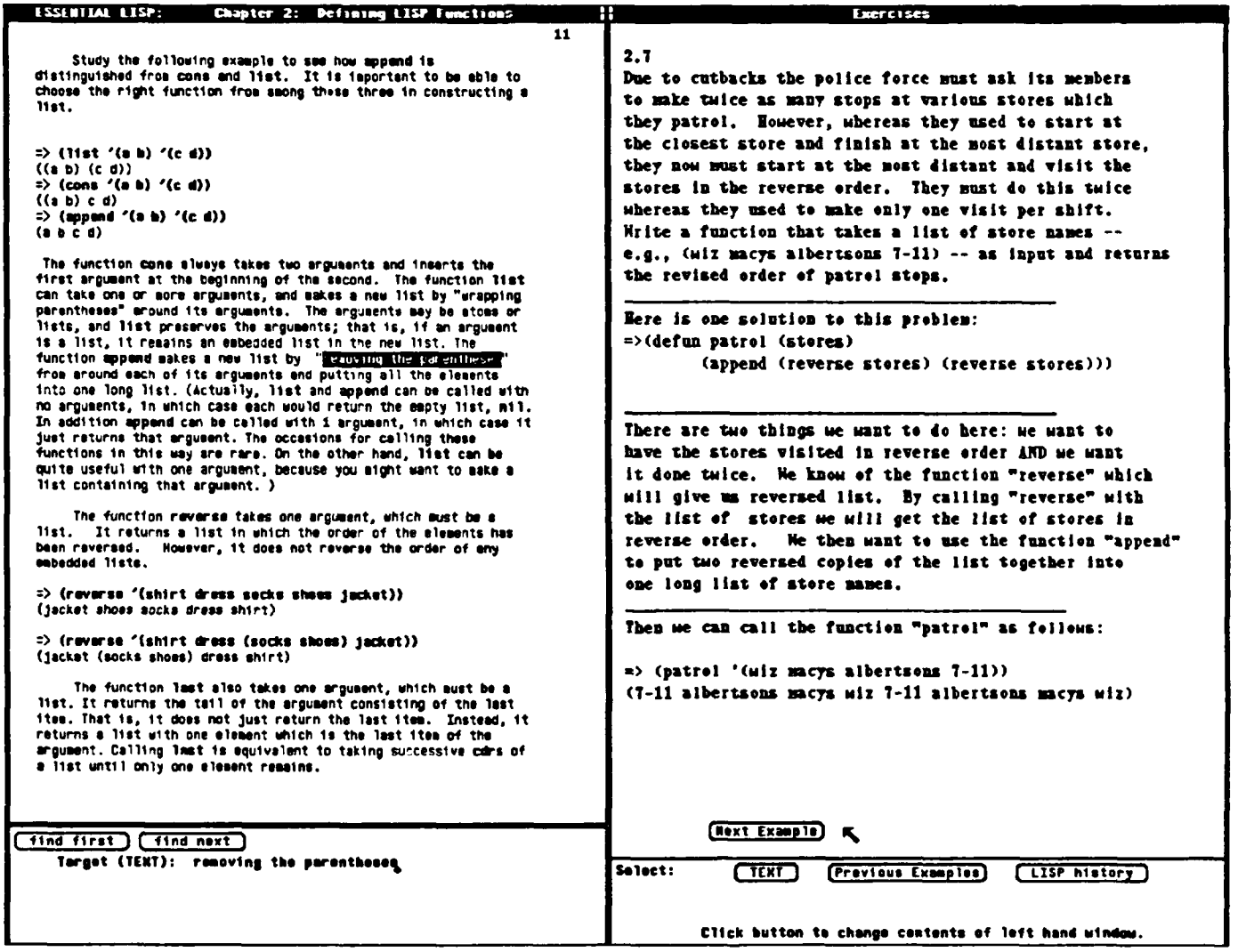


Figure 5. Reading a study example in the text.

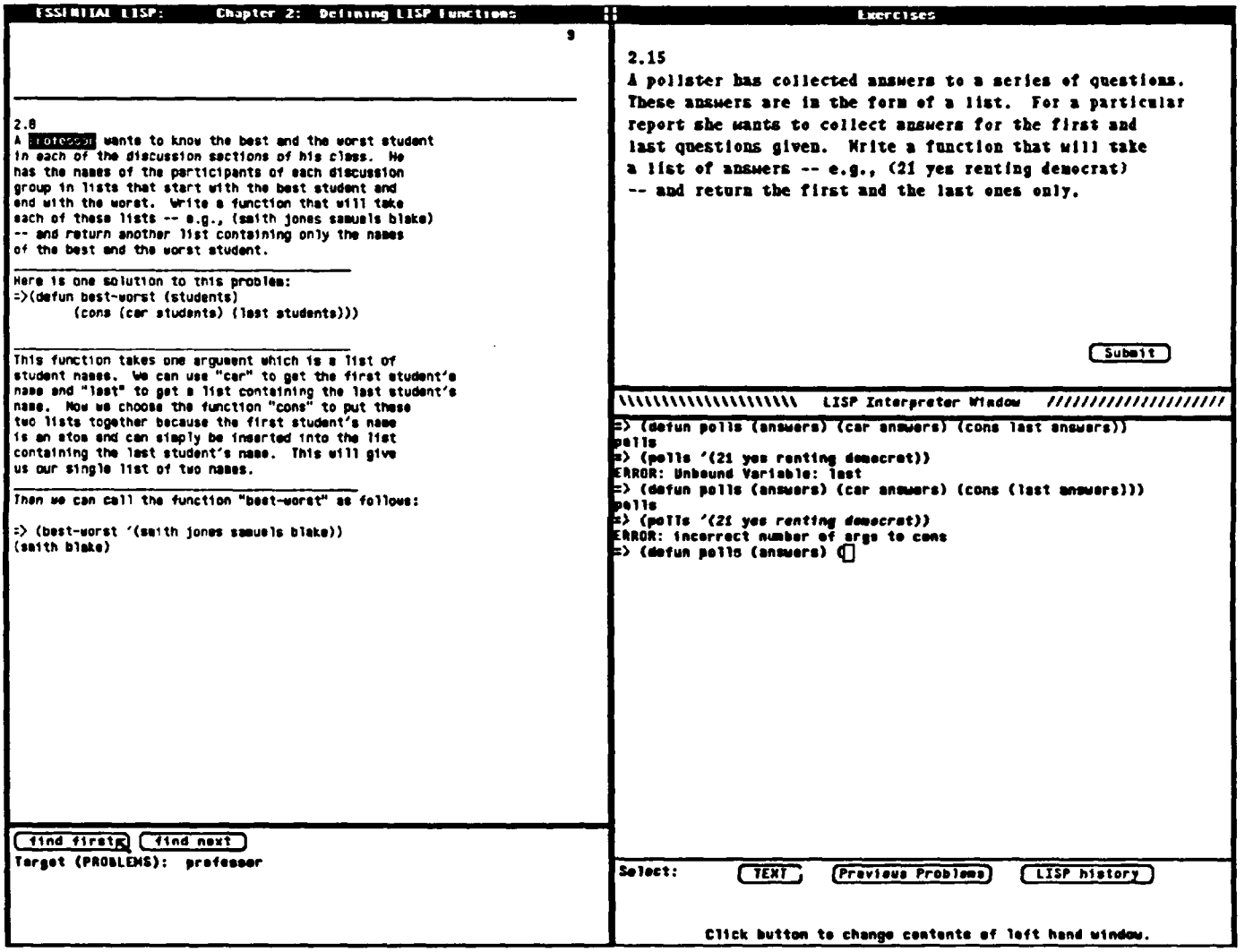


Figure 6. Retrieval of a study example from the text.

is displayed in the Text Window. If the search key cannot be found, a message is displayed and the student can either edit the search key or abandon the search. A case in which the student has retrieved a study example while solving a later problem is shown in Figure 6.

Making previous examples in the text more easily available to students is an important facility of the BATBook system. Many types of instructional text contain examples to demonstrate a concept or solution method. The successful "Minimal Manual" approach for computer manuals developed by Carroll and his colleagues relies heavily on the use of examples to demonstrate the sequence of actions to achieve a particular text editing goal (Black, Carroll, & McGuigan, 1987; Carroll, Smith-Kerker, Ford, & Mazur-Rimetz, 1987-1988). Students focus on examples in textbooks and manuals when learning new procedures (LeFevre & Dixon, 1986; Reder, Charney, & Morgan, 1986; Sweller & Cooper, 1985; VanLehn, 1986; Zhu & Simon, 1987). Furthermore, the type of elaborations students generate when understanding and reviewing examples strongly influences their success on later problems (Chi et al., 1989; Pirolli & Bielaczyc, 1989). Treating the example as a problem to be solved, in which each step can be "predicted" and the reasons for each step explained, leads to better learning. Consistent with these results, Charney and Reder (1986) demonstrated that students learn more by solving a corresponding problem than by simply following the step by step instructions in an example. The example searching capabilities of BATBook facilitate the process of learning from previous examples by enabling relevant examples to be more easily retrieved and used to solve new problems. As an experimental tool, the example searching capabilities of BATBook enable us to investigate when students decide to use previous examples and how they access them.

Another potential benefit of the use of examples in BATBook relies on the way in which examples are distinguished from regular expository text. This may serve to focus students on the examples and encourage them to process the examples more appropriately than treating them as additional expository text. Scardamalia and her colleagues have argued that making the understanding goals more explicit can encourage more profitable learning strategies (Scardamalia, Bereiter, McLean, Swallow, & Woodruff, 1989).

5.2 Searching Previous Solutions Using Labels

Students who store solutions using their own labels can later retrieve any problem and its solution using the label as a key. A search of previous solutions can occur at any point the student chooses. To search previous solutions, the student first selects the "Previous Problems" button in the LISP Interpreter Window. As in the

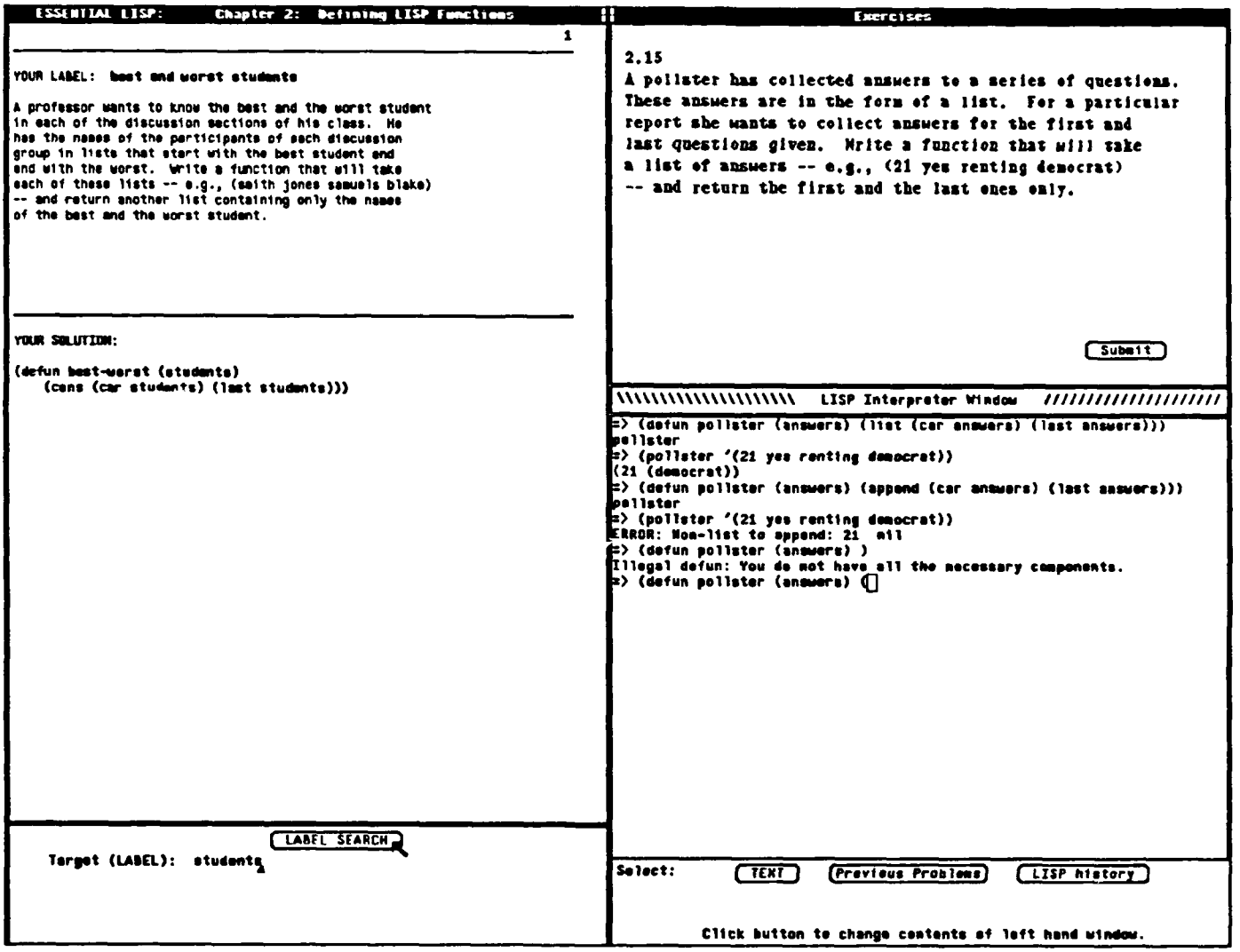


Figure 7. Retrieval of a previous solution using the student's label.

search of study examples, this replaces the Text Window with a Previous Solutions Window. The student can then type part or all of a previous label and select the "Label Search" button to initiate the search. If a problem with a matching label is found, the complete problem description and the subject's final solution to the problem are displayed in the Previous Solutions Window (see Figure 7). After finding a previous solution, the student can return to the LISP Interpreter Window to attempt to map a component of that solution to the current problem or decide to search for a different problem if the example retrieved turns out not to be useful.

Students find the ability to retrieve previous solutions a very useful feature of BATBook. The labeling facility provides pedagogical benefits similar to the facility for retrieving examples from the text. Much of problem solving early in a student's learning of a new topic relies on retrieving memories for previous solutions, adapting the procedure to fit a new problem, and generalizing the result. BATBook enables students to encode their solutions in ways that later can be used to retrieve them. The retrieval process is greatly facilitated — rather than having to recall a solution from memory, the computer enables students to accurately retrieve the complete problem description and solution. This type of assistance should facilitate the students' abstraction of general solution plans from the examples. This raises a number of important theoretical issues, which we are currently investigating using BATBook: What type of similarity causes a new problem to remind students of a previous solution? What aspects of a solution do students encode in memory?

There are potential pedagogical benefits in focusing students directly on the encoding and retrieval question. Students know when they label a solution that their labeling will affect how easy it will be to retrieve that solution at a later time. Requiring students to label their solutions may cause them to reflect on the solution and try to abstract what was important, interesting, or novel about this problem. This type of reflection is generally considered an important part of sophistication in a domain (e.g., Collins & Brown, 1988). BATBook provides a way to investigate these issues by examining students' choices of labels, whether the labels can be remembered and effectively used, whether the type of labels change with expertise, etc.

5.3 Searching Previous Solutions Using Problem or Solution Content

A different method for searching previous solutions is available to those students not asked to label their solutions when submitting them. These students can search by referring to any of the content of the problem or the solution. In this method, search is initiating by selecting the "Previous Problems" button. At that point, the student

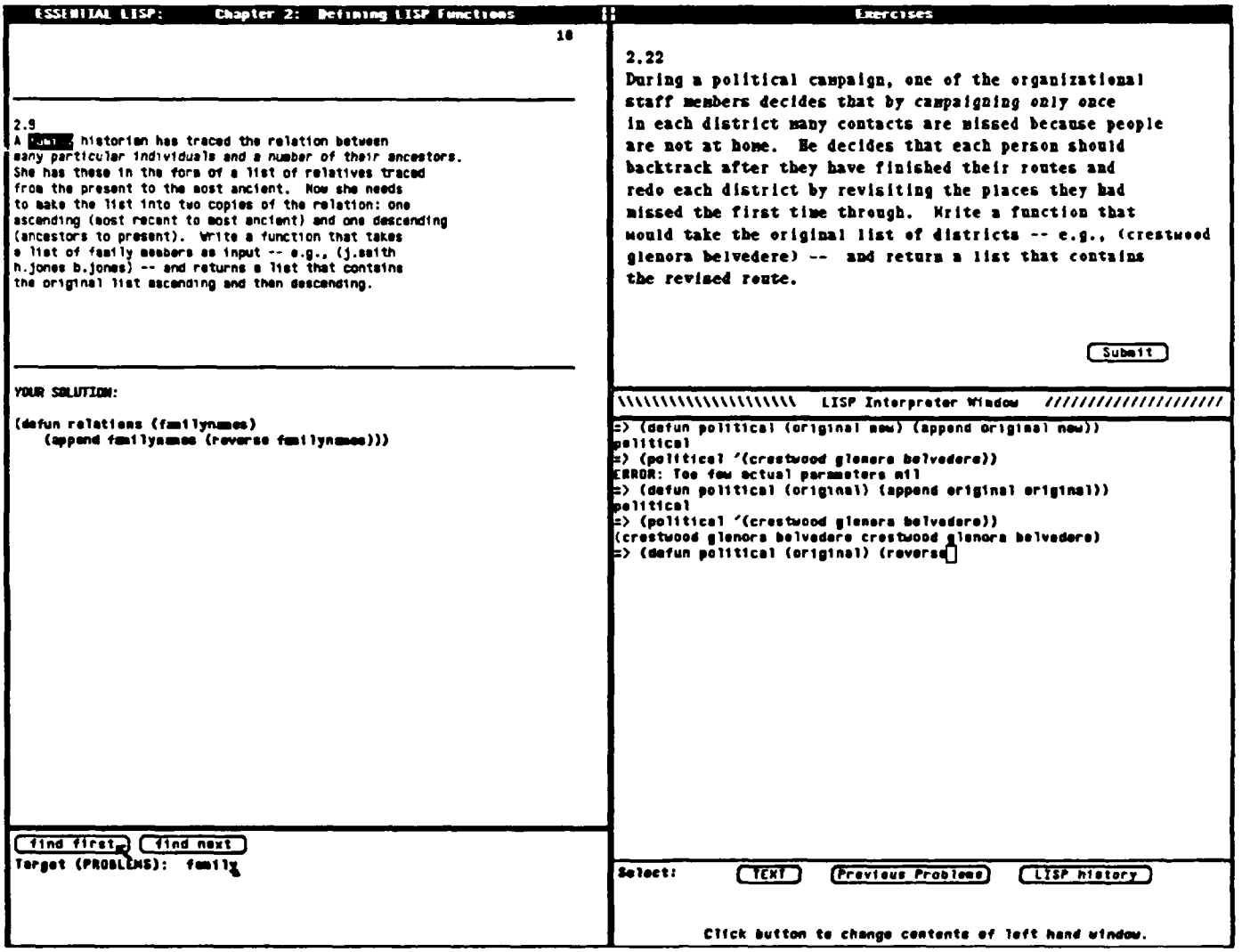


Figure 8. Retrieval of a previous solution using content of the problem.

can type in the search target, and select the "find first" or "find next" button. BATBook then searches through the problem descriptions and their solutions for the first or next occurrence of the search string. If a match is found, the problem description and the subject's solution are displayed in the Text Window, just as for a Label Search. Only the problem and description for a single problem are displayed at a time. An example of this type of search is shown in Figure 8.

5.4 Searching the LISP Interaction History

We have discussed how students can search study examples embedded in the text or their previous solutions. Another potentially important type of information is the problem solving attempts made prior to a solution. Information such as understanding an obstacle encountered, explanation of why a plan failed, or other unexpected events in pursuing a plan may play an important role in providing access to the event in memory and may determine what can be learned from the problem solving episode (e.g , Carbonell, 1986; Hammond, 1989; Seifert & Hammond, 1989). For this reason, BATBook provides students the ability not only to search complete examples or solutions but also to search the record of their problem solving interactions. In the LISP domain, this is accomplished by a facility for searching interactions with the LISP Interpreter.

Students can search the history of their interactions with the interpreter by selecting the "LISP History" button. At that point, the student can type a search string and select the "find first" or "find next" button. As in search of the text, this search finds the first or next occurrence of the search string in the log of the student's interaction with the LISP interpreter for that chapter. The log is displayed in the Text Window with the target string in reverse video. An example of this type of search is shown in Figure 9.

Students find the search of the LISP history to be a useful feature while solving problems. Students frequently search for previous episodes in which they encountered an error similar to one appearing in the current problem or for cases when they correctly used a construct relevant for a current problem.

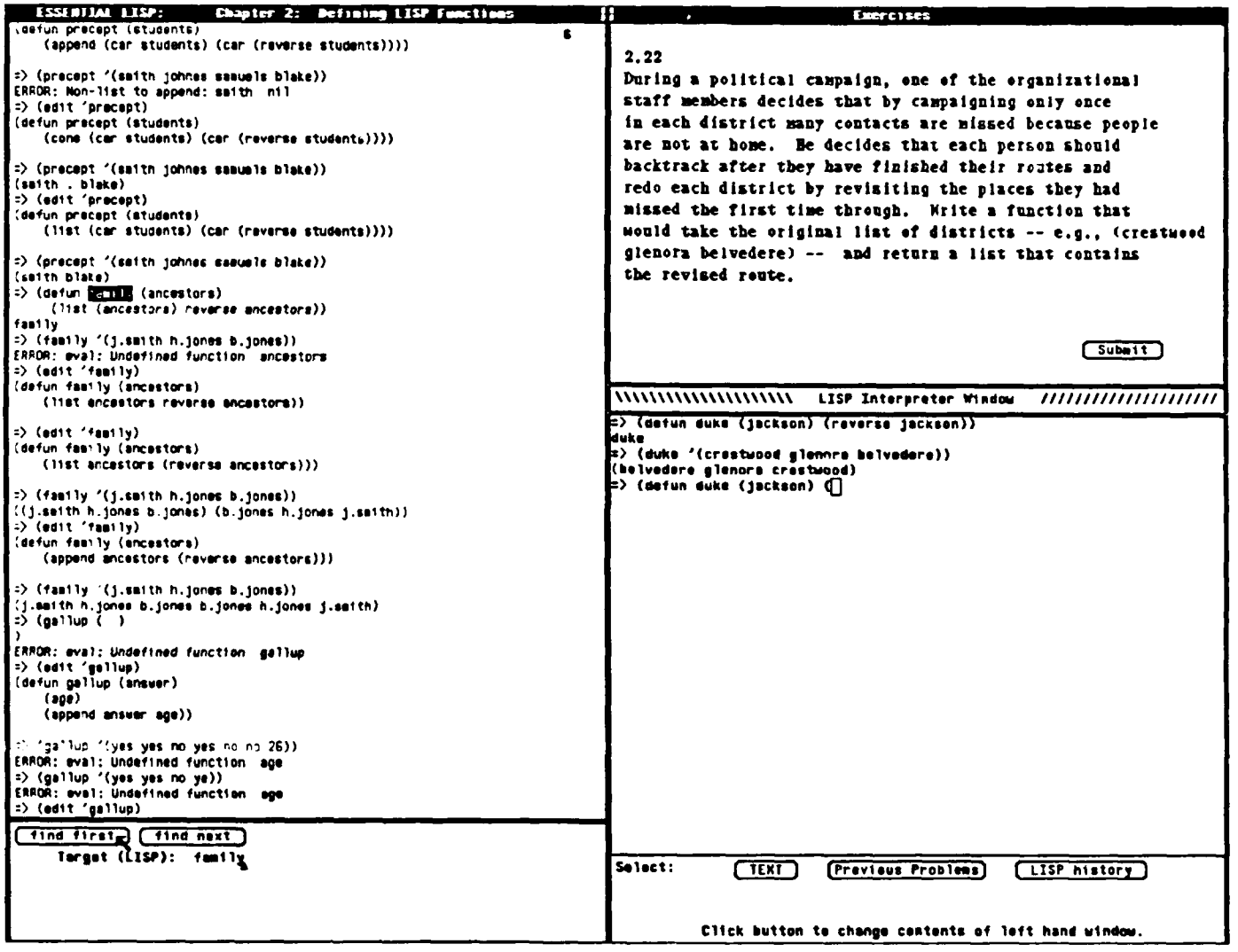


Figure 9. Retrieval of problem solving attempts in the LISP Interpreter Window.

6 Conclusions: The Study of Complex Cognitive Skills

In this paper, we have described the BATBook electronic book and problem solving environment. BATBook has been designed as both a pedagogical and an experimental tool. The pedagogical benefits of the system derive from the way in which BATBook facilitates students' use of examples. Study examples within the text can be easily retrieved and used when the student faces similar problems later on, or accessed in order to contrast a study example with a problem in which a somewhat different solution technique is required. The student's own completed solutions can be easily retrieved so that past successes can be applied to new problems. In this way, students can see what is similar between problems and abstract more general strategies and plans. Students can also retrieve a previous solution to contrast it with a current problem to consider why the previous technique cannot apply to the current problem. Furthermore, focusing students on the encoding and retrieval process itself may produce pedagogical benefits. When asked to label a solution, students must reflect on the problem solving episode and extract features useful for indexing it. In general, the problem solving environment encourages students to process examples differently from text, and to consider when solving a problem how it relates to other problems.

As an experimental tool, BATBook enables us to examine a number of important research questions facing theories of learning from examples. By making the students' use of examples overt and the process by which they retrieve examples explicit, we can examine in greater detail the role that examples play in learning. In particular, we are investigating the reminding and use of examples. What type of similarities between problems cause students to be reminded of a previous solution? Are students distracted by superficial similarities, or do they recognize structural similarities between problems with different surface content? At what point during problem solving do remindings occur? What type of information is encoded in memory from a problem solving episode?

We have constructed BATBook to examine the reminding and use of examples in a situation in which students are engaged in learning a new domain and solving problems. Retrieving a study example, an example of a bug encountered previously, or a successful solution are all methods in the student's battery of learning strategies. In this way, we can examine the use of examples in a natural context, where the examples are used as part of the problem solving. This approach differs from one in which the reminding is a goal explicitly posed by the experimenter, for example if the student were asked to describe similarities between problems or recall information that was only encoded for the purpose of recalling it. If the goal

is to characterize how people remember and use information to accomplish goals, it may be difficult to generalize from laboratory tasks in which the subjects are not given problem solving goals (Seifert, 1988; Seifert & Hammond, 1989). In order to investigate empirically the types of complex learning now facing cognitive science theories, it will be necessary to move beyond simple one-hour laboratory studies of simplified material with simple "percent correct" performance tests. Instead, much finer grained analyses will be required to evaluate learning models. Anderson (1987b) has argued that interactive problem solving environments that record and track students' reasoning are an optimal methodology for the study of learning. Experiments such as our BATBook studies (Faries, 1988; Faries & Reiser, 1988; Faries & Reiser, in preparation) and instructional experiments based on intelligent tutoring systems (e.g., Corbett & Anderson, 1989; Pirolli, 1986; Reiser, Ranney, Lovett, & Kimberg, 1989; Singley & Anderson, 1989) provide such intensive analyses of students' learning.

References

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review*, 89, 369-406.

Anderson, J. R. (1987a). Skill acquisition: Compilation of weak-method problem solutions. *Psychological Review*, 94, 192-210.

Anderson, J. R. (1987b). Methodologies for studying human knowledge. *Behavioral and Brain Sciences*, 10, 467-505.

Anderson, J. R., Corbett, A. T., & Reiser, B. J. (1987). *Essential LISP*. Reading, MA: Addison-Wesley.

Anderson, J. R., Farrell, R., & Sauers, R. (1984). Learning to program in LISP. *Cognitive Science*, 8, 87-129.

Black, J. B., Carroll, J. M., & McGuigan, S. M. (1987). What kind of minimal instruction manual is the most effective. *Proceedings of CHI+GI '87 Human Factors in Computing Systems*. New York: ACM, 159-162.

Carbonell, J. G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell, Eds., *Machine Learning, Vol. 2*, Los Altos, CA: Morgan Kaufman.

Carroll, J. M., Smith-Kerker, P. L., Ford, J. R., & Mazur-Rimetz, S. A. (1987-1988). The minimal manual. *Human-Computer Interaction*, 3, 123-153.

Charney, D. H. & Reder, L. M. (1986). Designing interactive tutorials for computer users. *Human-Computer Interaction*, 2, 297-317.

Chi, M. T. H., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13, 145-182.

Collins, A., & Brown, J. S. (1988). The computer as a tool for learning through reflection. In H. Mandl & A. Lesgold, Eds., *Learning issues for intelligent tutoring systems*. New York: Springer-Verlag.

Conklin, J. (Sept, 1987). Hypertext: An introduction and survey. *Computer*, 20, 17-41.

Corbett, A. T. & Anderson, J. R. (1989). Feedback timing and student control in the Lisp Intelligent Tutoring System. In D. Bierman, J. Breuker, & J. Sandberg, Eds. *Proceedings of the Fourth International Conference on Artificial Intelligence and Education*. Springfield, VA: IOS, 64-72.

Ericsson, K. A. & Simon, H. A. (1984). *Protocol analysis: Verbal reports as data*. Cambridge, MA: MIT Press.

Escott, J. A., & McCalla, G. I. (1988). Problem solving by analogy: A source of errors in novice LISP programming. *Proceedings of ITS-88: The International Conference on Intelligent Tutoring Systems*, Montreal, p. 312-319.

Faries, J. M. (1988) *Access and use of previous solutions in a problem solving situation.* Masters Thesis, Princeton University, Princeton, NJ.

Faries, J. M., & Reiser, B. J. (1988). Access and use of previous solutions in a problem solving situation. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, Montreal, p. 433-439.

Faries, J. M., & Reiser, B. J. (in preparation). The encoding and retrieval of problem solving episodes. Manuscript in preparation, Princeton University.

Furnas, G. W., Landauer, T. K., Gomez, L. M., & Dumais, S. T. (1984). Statistical semantics: Analysis of the potential performance of keyword information systems. In J. C. Thomas & M. L. Schneider, Eds., *Human factors in computer systems.* Hillsdale, NJ: Ablex.

Gentner, D. (1988). Analogical inference and analogical access. In A. Prieditis (Ed.), *Analogica.* Los Altos, CA: Morgan Kaufman.

Gentner, D. (1989). The mechanisms of analogical learning. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning.* New York: Cambridge University Press.

Halasz, F., Moran, T., & Trigg, R. (1987). Notecards in a nutshell. *Proceedings of CHI+GI '87 Human Factors in Computing Systems.* New York: ACM, 45-52.

Hammond, K. (1989). *Case-based planning: Viewing planning as a memory task.* San Diego, CA: Academic Press.

Holyoak, K., & Thagard, P. (in press). Analogical mapping by constraint satisfaction. *Cognitive Science.*

Kolodner, J. L. (1983). Towards an understanding of the role of experience in the evolution from novice to expert. *International Journal of Man-Machine Studies*, 19, 497-518.

LeFevre, J. & Dixon, P. (1986). Do written instructions need examples? *Cognition and Instruction*, 3, 1-30.

Lesk, M. E. (1975). Lex — A lexical analyzer generator. Computer Science Technical Report, No. 39, Bell Laboratories, Murry Hill, NJ.

Newell, A., & Simon, H. A. (1972). *Human problem solving.* Englewood Cliffs, NJ: Prentice-Hall.

Pirolli, P. (1986). A cognitive model and computer tutor for programming recursion. *Human-Computer Interaction*, 2, 319-355.

Pirolli, P., & Anderson, J. R. (1985). The role of learning from examples in the acquisition of recursive programming skills. *Canadian Journal of Psychology*, 39, 240-272.

Pirolli, P., & Bielaczyc, K. (1989). Empirical analyses of self-explanation and transfer in learning to program. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, Ann Arbor, MI, p. 450-457.

Reder, L. M., Charney, D. H., & Morgan, K. I. (1986). The role of elaborations in learning a skill from instructional text. *Memory and Cognition*, 14, 64-78.

Reed, S. K., Dempster, A., & Ettinger, M. (1985). Usefulness of analogous solutions for solving word problems. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 11, 106-125.

Reiser, B. J., Ranney, M., Lovett, M. C., & Kimberg, D. Y. (1989). Facilitating students' reasoning with causal explanations and visual representations. In D. Bierman, J. Breuker, & J. Sandberg, Eds. *Proceedings of the Fourth International Conference on Artificial Intelligence and Education.* Springfield, VA: IOS.

Remde, J. R., Gomez, L. M., & Landauer, T. K. (1987). Superbook: An automatic tool for information exploration — hypertext? *Proceedings of Hypertext '87*, p. 175-188.

Riesbeck, C. K., & Schank, R. C. (1989). *Inside case-based reasoning.* Hillsdale, NJ: Erlbaum.

Robertson, G., McCracken, D., & Newell, A. (1981). The ZOG approach to man-machine communication. International Journal of Man-Machine Studies, 14, 461-488.

Ross, B. H. (1984). Remindings and their effect in learning a cognitive skill. *Cognitive Psychology*, 16, 371-416.

Ross, B. H. (1987). This is like that: The use of earlier problems and the separation of similarity effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 13, 629-639.

Ross, B. H. (1989). Distinguishing types of superficial similarities: Different effects on the access and use of earlier problems. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15, 456-468.

Scardamalia, M., Bereiter, C., McLean, R. S., Swallow, J., & Woodruff, E. (1989). Computer-supported intentional learning environments. *Journal of Educational Computing Research*, 5, 51-68.

Seifert, C. M. (1988). Goals in reminding. *Proceedings of DARPA Workshop on Case Based Reasoning Workshop*, Clearwater Beach, FL, May 1989.

Seifert, C. M., & Hammond, K. J. (1989). Why there's no analogical transfer. *Proceedings: Case Based Reasoning Workshop,* DARPA Information Science and Technology Office, San Mateo, CA: Morgan-Kaufman.

Singley, M. K., & Anderson, J. R. (1989). *The transfer of cognitive skill.* Cambridge, MA: Harvard University Press.

Stallman, R. (1987). *GNU Emacs manual.* Cambridge, MA: Free Software Foundation, Inc.

Sweller, J. & Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, 2, 58-89.

Thagard, P., Holyoak, K., Nelson, G., & Gochfeld, D. (1989). Analog retrieval by constraint satisfaction. Unpublished manuscript, Princeton University.

Weyer, S. A. (1982). The design of a dynamic book for information search. *International Journal of Man-Machine Studies*, 17, 87-107.

Yankelovich, N., Meyrowitz, N. & van Dam, A. (October, 1985). Reading and writing the electronic book. *Computer*, 18, 15-30.

VanLehn, K. (1986). Arithmetic procedures are induced from examples. In J. Hiebert, Ed., *Conceptual and procedural knowledge: The case of mathematics*. Hillsdale, NJ: Erlbaum.

Zhu, X., & Simon, H. A. (1987). Learning mathematics from examples and by doing. *Cognition and Instruction*, 4, 137-166.