DOCUMENT RESUME

ED 309 407 CS 211 872

AUTHOR Neuwirth, Christine M.
TITLE Intelligent Tutoring Systems: Exploring Issues in Learning and Teaching Writing. CECE Technical Report No. 3.
PUB DATE Jun 88
NOTE 43p.
PUB TYPE Information Analyses (070) -- Reports - Descriptive (141)

EDRS PRICE MF01/PC02 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; Discovery Learning; Educational Technology; Higher Education; Program Descriptions; *Programed Tutoring; Technological Advancement; *Writing Instruction
IDENTIFIERS *Intelligent Tutoring Systems

ABSTRACT

This paper argues that Intelligent Tutoring Systems (ITS) offer the potential for advancing existing techniques for computer assisted instruction and deepening the understanding of issues in the learning and teaching of writing. The paper: (1) outlines the goals and significance of research in ITS, emphasizing their value in formulating knowledge about teaching more explicitly; (2) analyzes the pedagogical assumptions underlying research in ITS, focusing on theories of discovery learning and learning by doing; (3) describes the architecture of these systems; and (4) illustrates the instructional design issues by describing "Parnassus," a project to construct an ITS for exploring instructional design issues in teaching students to write. (Forty-five references and one figure are attached.) (RS)

Intelligent Tutoring Systems:

Exploring Issues in Learning and Teaching Writing

Christine M. Neuwirth

Center for Educational Computing in English
English Department
Carnegie Mellon University
Pittsburgh, PA 15213

Center for Educational Computing in English
CECE Technical Report No. 3
June 1988

ABSTRACT

This paper elaborates the potential and limitations of Intelligent Tutoring Systems in the teaching of language skills, in particular, of writing. It (1) outlines the goals and significance of research in Intelligent Tutoring Systems, emphasizing their value in formulating knowledge about teaching more explicitly; (2) analyzes the pedagogical assumptions underlying research in Intelligent Tutoring Systems, focusing on theories of discovery learning and learning by doing; (3) describes the architecture of these systems; and (4) illustrates the instructional design issues by describing PARNASSUS, a project to construct an Intelligent Tutoring System for exploring instructional design issues in teaching students to write.

BIO

Christine M. Neuwirth is an Assistant Professor of Rhetoric and Educational Computing at Carnegie Mellon University and Co-Director of CMU's Center for Educational Computing in English (CECE). Her research interests include rhetorical theory, especially the theory of argument, and the design and assessment of computer tools for writing. Her recent articles include "The Notes program: A hypertext application for writing from source texts," in *The Proceedings of the Hypertext '87 Workshop*, Nov. 13-16, 1987, Chapel Hill, NC, 121-141. With David S. Kaufer and Cheryl Geisler, she is co-author of the text, *Arguing from Sources: Exploring Issues through Reading and Writing*, Harcourt Brace Jovanovich, in press.

Introduction

Almost everyone has had the experience of trying to describe how to do something to someone else, only to discover that the knowledge is apparently exceedingly difficult to articulate. Sometimes procedural knowledge can be articulated, but because the description is so very complex, it seems that an easier, more efficient way to teach it is first to model the process for the student and then let the student practice while providing "tips." Much of the knowledge that writing teachers teach is of this nature. It resides in the heads of teachers, not necessarily in an articulable form; often the only, or at least the easiest, way to teach it is implicitly, by modeling the process and providing an environment for learning accompanied by advice on parts of the task.

Research in Intelligent Tutoring Systems begins with the premise that a detailed analysis of the processes required in performing a task can assist in the design of environments to teach it and that such work will take a step, albeit a small one, in furthering our understanding of the useful features of those environments (Collins, Brown & Newman, 1987). Developing techniques to formulate such knowledge explicitly offers a hope of being able to communicate more precisely and hence successfully--both among ourselves and with our students.

In this paper, I will elaborate the potential and limitations of Intelligent Tutoring Systems in the teaching of language skills, in particular, of writing. I will (1) outline the goals and significance of research in Intelligent Tutoring Systems; (2) analyze the pedagogical assumptions underlying such research; (3) describe the architecture of these systems; and (4) illustrate the instructional design issues by

describing PARNASSUS, a project to construct an Intelligent Tutoring System for exploring instructional design issues in teaching students to write.

Intelligent Tutoring Systems (ITS)

The term "Intelligent Tutoring System" (ITS) is widely used within cognitive science to refer to computer programs that can, supposedly, tutor intelligently. Strictly construed, however, the term ITS has no referent: no existing Intelligent Tutoring System exhibits the entire range of abilities we usually associate with the best human tutors. Despite the word "intelligent" in the name; and a *goal* of high performance, existing Intelligent Tutoring Systems are more modestly and commonly viewed as explorations into how computer technology can be used to specify or, more precisely, to formalize theories of tutoring, and as experimental vehicles for learning more about the nature of learning and teaching.

Goals of Research in ITS

Polya once observed (1957), "The first rule of teaching is to know what you are supposed to teach. The second rule of teaching is to know a little more than what you are supposed to teach." A cursory examination of the knowledge required by Polya's second rule (besides leading us to suspect Polya of litotes) can offer a glimpse of the diverse and complex goals of research in Intelligent Tutoring Systems. In addition to knowing what they are supposed to teach, tutors must know how to carry out the following activities:

Construct a model of the student. While observing a student solving a problem, tutors must be able to abstract general processes from the student's behavior and infer the student's strengths and weaknesses, perhaps strengths and weaknesses they have never seen before. To decide which features of a student's behavior are significant, tutors must be guided by a theory, often implicit, of the underlying skills and concepts necessary for success. Both to refine their understanding and to monitor a student's progress, tutors must be able to test and revise their hypotheses about what a student knows.

Structure an environment for learning. Based on a model of the student, an understanding of the domain, an agenda for instruction, and tutorial strategies, tutors must choose or create appropriate problems for the student to work on. They must decide on problem-types, what sequence to present them in, and how many of each to present. They must also be able to adjust to different students' backgrounds, introduce or review material as the need arises, focus the student's attention, explain the task, give hints to the student about how to solve a problem, and provide model solutions.

Interact with the student. Tutors are not the only teachers who structure an environment for learning. Textbook writers and lecturers must, of course, also do so, though without as much sensitivity to individual students and their immediate behavior as tutors can achieve. Tutors often interact with a student while the student is solving a problem. They must decide when to interrupt a student's problem-solving activity and what to say when they do interrupt. They must be able to give advice within the student's conceptualization of a problem, perhaps

prompting the student with the next step to take; they must be able to instruct in alternative conceptualizations, perhaps pointing out better ones; to interact most effectively, they must be able to recognize when an able student's conceptualization is better than their own. Tutors must also be able to participate in a dialogue with a student, responding to the student's initiative, providing requested information, perhaps solving new problems that the student creates. To gain greater expertise, tutors must be able to learn new strategies for tutoring by reasoning about the interaction itself.

By attempting to build computer programs that can do the things human tutors can do, researchers in ITS further our understanding of the knowledge and processes that make intelligence possible. Although each existing ITS has concentrated on some aspect of intelligent tutoring and neglected others, all the systems have advanced our understanding of what it is that tutors know, including both knowledge of the subject matter and knowledge of how to teach it. Building an Intelligent Tutoring System requires formulating such knowledge explicitly and comprehensively. Two benefits result from a more explicit formulation of knowledge. First, other theorists have something specific to challenge or revise. Even when the initial formulations are based on *ad hoc* or unprincipled intuitions to bring about desired behavior, they can serve to reveal gaps in our understanding and spur the development of more detailed, psychologically validated theories. Second, because programs can be systematically modified, work in Intelligent Tutoring Systems can be a testing ground for experimenting with alternative theories.

Pedagogical Assumptions Underlying ITS: Why Tutor?

The word "tutor" can be traced to the Latin word *tutus*, a protector. The connotations of "keeping safe," "watching over," and "promoting growth" evoked by "protector" persist in modern usage, especially when contrasted with "lecturer." Beyond its denotation as "a person charged to instruct another, especially privately," the word "tutor" can convey a loosely related set of assumptions concerning the way a teacher *ought* to instruct. All research in Intelligent Tutoring Systems shares a definition of the role of a teacher as a tutor rather than lecturer, but differs in the assumptions made about how one ought to tutor. These differing assumptions stem from researchers' designs being primarily motivated by one of two concepts: "discovery learning" or "learning by doing." Both concepts can motivate tutoring, but the two concepts encourage different methods. Of course, other theories of learning and instruction could motivate tutoring as well.

Discovery Learning

An assumption shared by many existing Intelligent Tutoring Systems is that it is best for students to discover for themselves as much of the structure of a subject as possible. Known as "discovery learning," this assumption can be traced to ideas developed in the early 1900s by such educational leaders as J.J. Rousseau, Maria Montessori, and John Dewey (Wittrock, 1966). The assumption can also be seen in learning theories developed by Gestalt psychologists such as Wertheimer (1959). In *Productive Thinking*, Wertheimer challenged methods of

teaching that emphasized rote performance and recall: such teaching methods often failed to produce understanding and insight, resulting instead in "senseless combinations." Instead, Wertheimer advocated a Gestalt approach to teaching, in which students would learn from more holistic and meaningful tasks, finding and structuring problems and solutions for themselves, and discovering general principles. Related studies which lend support to Wertheimer's position on rote learning suggest that material learned with understanding is learned more rapidly, retained over longer periods of time, and transfers better to new tasks than material learned by rote (Katona, 1940, chap. 4).

Gestalt ideas have influenced the study of learning and memory in recent cognitive psychology. For example, the question raised by discovery learning of *how much* understanding students should achieve by themselves is now coming under systematic investigation (Charney & Reder, 1986; Lewis & Anderson, 1985), though many questions remain. Nevertheless, an analysis of what discovery learning requires from a student can provide the basis for discussing its *prima facie* attractions and problems. In discovery learning, a student must work on a series of meaningful tasks, detecting problems, analyzing the task for features that might be relevant to a solution, and formulating goals and subgoals that will contribute to solutions. The student must also systematically test a solution for possible errors. In order for the student to detect and correct a misconception, the misconception must cause an erroneous or problematic result, which the student must notice. The student must correctly hypothesize the cause of the problematic result and be able to correct it. In short, discovery

learning requires a student to exercise the processes involved in original inquiry, learning not only knowledge, but also the knowledge of how to acquire knowledge, that is, "learning to learn."

Exercising the processes involved in "learning to learn" makes discovery learning attractive and, at the same time, is the basis for its problems, for students vary substantially in how good they are at discovery. A task which is meaningful may be too difficult for a student to do without help. A student may form misconceptions that are never detected. A student may fail to correct a misconception.

The problems students have with discovery learning, together with the assumption that they indeed learn best by discovery, encourages a definition of the role of the teacher as a tutor: a teacher should not lecture, but rather should monitor a student's discovery learning, providing guidance when appropriate and adapting that guidance to each individual student's varying ability to learn. Furthermore, the guidance should be minimal. If students are to exercise processes of discovery, that is, learn how to learn, then providing too much guidance may prevent students from developing the skills of examining their own behavior and looking for the causes of their own mistakes. Unfortunately, there are no precise principles for how to guide a student's discovery minimally, though there are some general guidelines. For example, an ITS built according to "discovery learning" guidelines might not give a student immediate feedback on an error; instead, it might pick tasks for the student with salient features which will increase the likelihood that the student will notice the error (cf. Burton &

Brown, 1979).

Discovery learning is a theory of instruction, and as such, *prescribes* blurring the boundaries between learning things already known to others and learning things that are unknown. In contrast, the related concept of "learning by doing" is a theory of learning, that is, a descriptive theory of *how* students learn rather than how they *ought* to learn. As we shall see, however, it also blurs the boundaries between learning and discovery, at least for knowledge of skills.

Learning by Doing

Exploring the concept of "learning by doing" requires a theory of skill acquisition, where "skill" simply means any activity that a person, either from talent or training, has learned to do well, for example, playing piano, sweeping a floor, composing poetry, remembering people's names, and so forth. One of the most precise and well-developed psychological theories of skill acquisition is that of ACT* (Anderson, 1983), and it will serve as a basis for my discussion.

Central to the ACT* theory is the concept of a production system, an interrelated set of condition-action pairs, called "productions." The conditional part of a production specifies a set of data patterns. If data in a structure called "working memory" matches the conditions specified by the data patterns, then the action part of the production can be taken. Less technically, a production specifies what to do in a given, perhaps complex condition. Productions systems are usually traced to the proposals of Post (1943), with modern production systems diverging on the one hand into computer science applications, in which the focus is on

creating an architecture for engineering computer programs, and on the other into psychological applications, in which the focus is on creating an architecture that can account for and predict human behavior. Informally, a production might look like the following:

> IF the goal is to fix a topic-shift problem in a sentence
>
> and the problematic sentence shares a referent with the previous sentence,
>
> THEN set a goal to subordinate the problematic sentence to the previous sentence.

A production system model of an expert writer would contain hundreds of such productions, each production working with the others to produce expert writing processes. For example, the production above might be one production in a model of expert writers' revision processes. The model would include other productions for detecting a topic-shift problem and alternative productions for correcting it (Hayes, Flower, Schriver, Stratman & Carey, 1985).

The ACT* production system, a psychological application, posits three structural components to human cognition:

Declarative memory: Knowledge of facts and their relations, often called declarative knowledge. It represents permanent or long-term memory, and the units represented can take the form of propositions, temporal strings, or spatial images. For example, a fact such as "A canary is flying over the tree" could be represented by a proposition (e.g., [(AGENT canary) (RELATION fly)

(LOCATION [(AGENT canary) (RELATION over) (OBJECT tree)]]), by a temporal string (e.g., "A canary is flying over the tree"), or by a spatial image (e.g., an image of a canary flying over a tree). Knowledge in declarative memory is mediated through processes of storing elements from working memory and retrieving elements into it.

Working memory: Declarative knowledge that is active, either permanent knowledge retrieved from declarative memory, or temporary knowledge deposited by processes which encode information from the environment, or by actions of productions.

Production memory: Knowledge of how to do things, often called procedural knowledge. The productions operate on the declarative knowledge, mediated through working memory by matching and execution processes.

In ACT*, the acquisition of skill begins with an interpretive stage in which domain-independent productions, or general problem-solving productions, access a declarative representation of the skill. The declarative representation could come from textbooks, lectures, a tutor telling a student how to do a task, or a student observing an expert doing the task. This declarative representation, coupled with general interpretive processes, is sufficient to approximate the skill, but only crudely and laboriously. To achieve skill, new task-specific productions must be built. New productions are built automatically, in response to practice, from a memory trace of production application. These new productions increase the efficiency of performance by eliminating the stage in which declarative

knowledge is retrieved from long-term memory and interpreted, and by composing sequences of productions into new ones with fewer intermediate steps. At first, these productions diverge substantially from expert productions, and a student makes many errors. A process of fine tuning, generalizing or discriminating the conditions of existing productions, must occur.

The above discussion presents only a general framework of the ACT* theory, which is much richer than sketched here. Although the framework presented is not very detailed, it is complete enough to make several predictions about how students learn, predictions which correspond to researchers' observations of the development of skilled performance (cf., Anderson, 1983): some knowledge, knowledge of facts, can be learned well through textbooks, lectures, or verbal instruction, that is, instruction that results in a declarative representation; in contrast, knowledge of how to do things, procedural knowledge, can only be learned well through "learning by doing." Textbooks, lectures, and verbal instructions can play a role in the acquisition of skill, but they do not suffice. To achieve excellence, a learner must create new productions and discover refinements of old ones, processes which can only occur when productions are actually applied, that is, by doing a task. Furthermore, since the productions built by doing a task are task specific, it is important that students practice by actually doing the task or a closely related one.

Since learning by doing requires students to create new productions and to discover refinements of old ones, not surprisingly, it shares a basic problem with discovery learning: students vary substantially in how good they are at discovery.

A complex task may be too difficult for a student to do without help. A student may practice errors and ineffective methods.

Although "learning by doing" is a theory of learning and as such, does not prescribe a particular method of teaching, Anderson, Boyle, Farrell & Reiser (1984) have outlined a series of principles for instruction which are derived from the ACT* theory of learning. These include communicating expert goals to the student, providing instruction *during* problem-solving, giving immediate feedback on errors, minimizing the student's working memory load, using production system models of what the student knows and does not know to gauge appropriate instruction, adjusting the instruction with estimates of learning, tutoring in problem contexts that are as similar to the target skill as possible, and emphasizing a general problem-solving approach. Anderson and his colleagues have followed these principles in the design of two Intelligent Tutoring Systems, one for teaching the programming language LISP, and the other for teaching geometry.

Thus, learning by doing also encourages a redefinition of the role of a teacher as a tutor: rather than just telling a student how to do a task, a teacher should monitor the student's practice, providing guidance when appropriate. The methods, however, differ from those encouraged by discovery learning. For example, in learning by doing a tutor can tell a student how to structure a problem or can give the student immediate feedback on error.

In sum, all previous work in Intelligent Tutoring Systems has focused on tutoring in procedural skills, either in subjects which are inherently procedural or in subjects which deal with facts, but where the instructional objective is for students to learn how to acquire facts or use them as a basis for reasoning. Unfortunately, there are no precise rules for how a teacher ought to structure an environment in which to guide student's discovery, though certain teachers have, through experience, developed a knack, and attempts have been made (Collins, 1975) to define their methods more precisely. The architecture of Intelligent Tutoring Systems, however, demand a very precise formulation of method. The particular methods which have been developed for ITS differ depending on whether the researcher subscribes to the monitoring suggested in the concept of learning by doing, or that suggested by the concept of discovery learning. Which approach, or combination of approaches, would lead to the best long-term gains is an open research question, a question that has taken on renewed interest and greater precision as researchers build Intelligent Tutoring Systems.

ITS Architecture

Intelligence in computer programs, as in people, is a matter of degree. The following section surveys past work in computer-assisted instruction (CAI), focusing on the advances in the underlying architecture that can be viewed as steps toward more intelligence. As O'Shea and Self (1983) observe, these advances can all be seen as advances in representations of what is being taught (the expert model), who is being taught (the student model), and how to teach it (tutorial strategies).

The Expert Model

Early drill and practice programs contained only an implicit representation of the subject matter to be taught. For example, a CAI grammar drill and practice program may represent grammatical knowledge in program statements that print information to the computer screen (e.g., PRINT "If you are trying to fix a sentence with a topic-shift error, try subordinating it.") When the representation is implicit, however, other parts of a computer system cannot access or interpret the information. With an explicit representation of knowledge (e.g., IF the goal is to fix a topic-shift problem in a sentence and the problematic sentence shares a referent with the previous sentence, THEN set a goal to subordinate the problematic sentence to the previous sentence) a computer program can be written so that different components can use the knowledge for multiple purposes (e.g., to generate solutions to topic-shift problems, to generate advice for students about how to solve such problems). Explicit representation of knowledge was the first step in developing more intelligent programs and was first used for "generative CAI" programs which, by examining a representation of knowledge, can generate new problems and questions (Laubsch, 1975).

Explicit representation of knowledge has taken two forms, declarative and procedural. The first use of an explicit declarative representation for representing subject-matter expertise in tutoring was by Carbonell (1970) in the SCHOLAR program, which used the representation as the basis for its ability to generate and answer questions in geography. Explicit representation of declarative knowledge has been argued to be essential in language understanding and generation

(Schank, 1975; Schank & Abelson, 1977; Schank & Riesbeck, 1981). Procedural knowledge of the subject matter is required if the program is going to be able to show the student how to do things. Numerous Intelligent Tutoring Systems have represented procedural knowledge as a production system (e.g., Clancey, 1979; Brown & Van Lehn, 1980).

Much recent research in writing has been concerned with observing expert writers at work and building models of expert processes (e.g., Hayes & Flower, 1980). Although these models are rather specific about the nature of expert writing processes and their organization, considerable work remains before the models are specific enough to be used to build an ITS.

The Student Model

A student model is a representation of a student that attempts to account for what the student knows and needs to know. For example, to represent that a student is weak in detecting topic-shifts, weak in fixing topic-shifts, and strong in punctuating correctly, a student model might, informally, look like the following:

(STUDENT <name of student> ((DETECT TOPIC-SHIFT) WEAK) ((FIX TOPIC-SHIFT) WEAK) ((GENERATE PUNCTUATION) STRONG))

Student models are typically used to adapt instruction to the student.

When early CAI programs modeled the student at all, they used mathematical models. Mathematical models often represent learning probabilistically. For example, a mathematical model of vocabulary learning might hypothesize that

each vocabulary word is in one of three states (learned and will not be forgotten between sessions, learned but might be forgotten between sessions with probability p, or unlearned). The model might further specify the probabilities of a word being in any state immediately after a vocabulary learning session, based on other probabilities, such as the probability of the student attending to the word during the session. Mathematical models, like all representational systems, have both strengths and weaknesses. Mathematics is a good vehicle for precise statement and economical abstraction. In its most successful instances, it can be used to predict a complex network of relationships in data from a very simple conception of laws governing behavior. However, abstract mathematical models can describe and predict data without having either a psychological rationale or a clear identification of the predicted data with processes. Therefore, the trend in modeling has been away from an emphasis on mathematical models that simply describe data toward a description of the processes which suffice to produce the described behavior (O'Shea & Self, 1983, pp. 83-86).

Two such approaches to modeling a student have been developed. The first, the expert-subset approach, models the student as knowing or not knowing what the expert knows, usually with some estimate of probability (e.g., Carbonell, 1970; Clancey, 1979). Since the student's understanding is represented completely in terms of the expert component of the system, the model assumes that the student reasons in the same way as the expert, but simply knows less. Another approach is to model the student's processes themselves, usually organized as deviations from expert rules (Brown & Van Lehn, 1980), or as evolutions in the

acquisition of skill (Goldstein, 1982). This allows a representation of the student that can differ more substantially from the expert, but how to represent deviations or evolutions--and diagnose and generate them--is a major research problem.

Student models of writers can take either an expert-subset or a model of a student's own processes. Within the expert-subset approach, it is easy to model a student who fails to use some of the expert processes, for example, to model a student who fails to organize an essay. Student writers' evolutions in the acquisition of skill is also coming under systematic investigation (Bereiter & Scardamalia, 1987).

Tutorial Strategies

Early computer-assisted instruction programs did not have explicit representations of tutorial strategies. Instead, the author pre-specified the material to be presented and what should be presented next. Usually, the only information about the student taken into account by the author-specified strategy was the last response and a percentage of questions answered correctly.

Recent work in Intelligent Tutoring Systems can be characterized as an attempt to make tutorial strategies explicit and precise enough to be programmed. The most well-articulated theory of tutoring is embodied in Collins (1975) and Stevens, Collins & Goldin (1982). It represents an attempt, based on a study of human tutorial interactions, to formalize the Socratic method. Other theories, much more experimental and less well generalized, have developed out of the "discovery learning" paradigm, with its emphasis on minimal intervention. (Burton

& Brown, 1979; Norman, Gentner, & Stevens, 1976). This is a relatively new direction and not as well developed as the work by Collins. However, the resulting dialogues do give insight into strategies a language tutor might employ. Ohlsson presents a cogent argument for the centrality of research on teaching methods to the construction of ITS (Ohlsson, 1987).

Tutorial strategies concern decisions of what to say and when to say it. Actually saying things involves strategies for discourse production. Work on integrating tutorial strategies with principles of discourse production began with Clancey (1979). Clancey developed a set of production rules to ensure smooth transitions between discourse topics and to adapt dialogues to a student's knowledge. Woolf & McDonald (1984) have also designed ways to represent discourse strategies for tutorial dialogues.

PARNASSUS: An ITS for Writing

The purpose of the PARNASSUS project is to construct an Intelligent Tutoring System to explore issues of learning and teaching in writing. The design of the system raised many of the issues outlined above, and its discussion will serve as an illustration of the instructional design issues facing anyone seeking to construct Intelligent Tutoring Systems for language instruction.

ITS and the Teaching of Writing

Research in Intelligent Tutoring Systems draws heavily on its parent discipline, Artificial Intelligence. Techniques in Artificial Intelligence have not yet progressed

to the point where a computer can be programmed to write, much less intelligently teach someone how to write. The best efforts at programming computers to *generate* discourse (Mann & Moore, 1981; McKeown, 1985) are restricted to limited topics, and although they are extremely impressive from the standpoint of research in computer science, writing teachers would hardly want their students to imitate the prose they can produce. Similarly, efforts at programming a computer to *understand* discourse suffer the same limitations (Schank & Riesbeck, 1981): understanding is limited to small, well-defined topics, and the prose that can be comprehended is best characterized as "primer prose."

Given the current "state of the art" in AI, it is not surprising that research in intelligent language systems does not set out to program computers to teach writing. Researchers must focus attention on some subset of the process of writing. Finding such a focus requires searching for a significant writing task that is programmable on the computer, yet pedagogically successful, and adapting that task, if necessary, to the limitations of state-of- the-art technology. The particular focus of the PARNASSUS project is teaching students to write effective sentences, defined as sentences that are appropriate to the context in which they are written.

A Revision Paradigm

Before discussing each element of the tutorial interaction in detail and the design issues it raises, I will outline a tutorial interaction with a prototype version of the

PARNASSUS system.[1] Figure 1 depicts a typical exercise. The exercise is based on a revision paradigm. The paragraph in Figure 1 illustrates one type of ineffectiveness, that is, inappropriateness to context: a violation of the norm of paragraph unity, often called "global coherence." Note that all the sentences in the paragraph cohere locally or are cohesive (Halliday & Hasan, 1976) but that the last sentence in the innermost box introduces a new idea ("the many ways that humanity has advanced") and is not related to the topic of the paragraph as a whole ("humanity's fortitude"). Thus, the paragraph lacks global coherence.

The program asks the student to read a paragraph, then asks the student to revise the boxed sentences so that they are more effective. Using a word processor, the student produces a revision. The program compares the student's revision to a set of possible revisions. If there is a better revision and an inspection of the student model indicates that the student needs instruction in the rhetorical goals illustrated by the better revision, the program informs the student that it has a revision that it has rated better and asks the student whether he or she wants to try to produce it.

The Task

Because constructing an Intelligent Tutoring System requires substantial resources, it is important that the task be chosen carefully if the system is to have practical as well as theoretical and research value. That lack of coherence is a major problem in student's writing is well-documented (cf., Bamberg, 1984). Moreover, work by Freedman (1979) suggests that organizational coherence

contributes significantly to judgments of writing quality.

It is the goal of the PARNASSUS project to create tasks that require students to "learn by doing." For example, the revision task just outlined requires students actually to *produce* effective sentences. Because of the difficulties involved in processing natural language text, traditional CAI programs often require students to do activities that do not require them to produce language (e.g., to *choose* the most effective sentence from a list of sentences, to *learn about* syntactic structures such as appositives or relative clauses that comprise effective sentences, etc.).

According to "learning by doing" theories, it is important that students practice by actually doing the task or a closely related one. The task that we want students to learn is how to produce effective sentences, i.e., sentences that are appropriate to the context in which they are written. In order to arrive at an exercise format that was achievable by the "state of the art" in AI, we made two design choices that potentially compromise the "learning by doing" goal. First, rather than a free-form sentence production task, we chose a revision task so that students' responses would be constrained somewhat. Second, the exercises are designed so that syntactic strategies suffice to fix the problems. Both design choices simplify the program's processing of students' responses. Note that although syntactic strategies suffice to fix the problems, the program does not focus on syntactic strategies for their own sake. Instead, the focus is on the underlying rhetorical goals which might prompt a writer to make syntactic changes, that is, on the *function* of syntax (e.g., maintaining global coherence,

emphasis, etc.). Based on an analysis of the skills involved, we predict that there will be transfer to the production of effective sentences and improvement in writing quality (Neuwirth & Ogura, 1987). Of course, whether such transfer actually occurs is an empirical question.

The task is a simplification of the task that we would ultimately like to program: in our simple prototype program, the student is *told* that the boxed sentences are not maximally effective; we would like to expand the task so that the student must also *detect* the problem (Flower, Hayes, Carey, Schriver & Stratman, 1986). In such an expanded version of the task, the program would ask the student to read a paragraph, then ask the student whether there were any ineffective sentences in the paragraph. Thus, students would be able to exercise detection as well as correction skills.

The Expert Model

As an example of what the expert component must do, consider the sentences "Humanity has hardly advanced in fortitude since that time" and "It has, however, advanced in many other ways." These sentences are part of the larger paragraph presented as a sample exercise. Using heuristics formalized as a set of productions, the expert component must generate a large number of possible revisions including, but not limited to, the following sentences:

(1) Although humanity has hardly advanced in fortitude since that time, it has advanced in many other ways.

(2) Although humanity has advanced in many other ways since that time, it has hardly advanced in fortitude.

(3) Humanity has hardly advanced in fortitude since that time; however, it has advanced in many other ways.

(4) Humanity has hardly advanced in fortitude since that time, but it has advanced in many other ways.

Informally, a partial, example set of productions for generating revisions is given by the following:

IF the goal is to fix a topic-shift problem in a sentence
and the problematic sentence shares a referent with the previous sentence,
THEN set a goal to subordinate the problematic sentence to the previous sentence.

IF the goal is to subordinate sentence B to sentence A
and sentence A and sentence B share a referent
THEN set a goal to create a relative clause.

IF the goal is to subordinate sentence B to sentence A
and sentence A and sentence B are related by the relation 'contrast'
THEN set a goal to use a contrastive conjunction.

These productions, which provide an explicit representation of the knowledge needed to carry out the revision task, produce a set of sentences. Some of the sentences produced are good, some not. A rating routine must determine which of the sentences produced by the productions is the best, next best, and so forth. This rating is implemented by comparing a representation of the sentence in terms of the goals that it fulfills (e.g., global coherence, emphasis, grammaticality) to a model of the goals required in a current discourse context (McKeown, 1985). The comparison involves an ordering of goals according to their importance, and assigning a higher ranking for matching a more important goal. This ranking is also adjusted to favor sentences which satisfy more goals. In the current system, sentence (2) is rated highest; sentences (1), (3), and (4) are rated the same as the original.

Most of the work in PARNASSUS has gone to creating a production system for generating revised sentences, given a description of the context and and an identification of a problem. A great deal more work must be done. Because the production system in the prototype PARNASSUS is incomplete, a set of possible revisions, both good and bad, is fed into the program.

Interestingly, a set of productions for producing revisions is only a subset of those necessary for carrying out the revision task. A more complete set would include productions that assess the contribution of each sentence to the topic of the paragraph and productions that could detect ineffective sentences. It would be useful to have a more complete set, since it would help to form a better model of the student.

Understanding and judging students' responses

Rather than generating both good and bad sentences and rating them, the expert component could have been built so that it only generated the most appropriate sentences, but that approach would complicate the problem of processing the students' responses.

Students' responses can vary widely. For example, students responses to the exercise in Figure 1 could include "Although humanity has advanced in many other ways since that time, it has not advanced in fortitude," "Humanity has not advanced in fortitude, but it has advanced in many other ways," "Humanity has made many advances since that time, but it has not advanced in fortitude," and so forth. The tutoring component must be able to judge a wide range of student responses. A student's revision may contain grammatical errors. For example, the punctuation error "Humanity has hardly advanced in fortitude since that time, however, it has advanced in many other ways."

As these examples indicate, the crux of the tutoring problem in writing is to be able to handle the students' responses: students often produce ungrammatical responses which the system must be able not only to detect, diagnose, and perhaps remediate, but also to ignore, at least temporarily, should another feature have higher priority for tutoring. Of course, the system could be built so that grammatical errors were not temporarily ignored, but it would not be flexible enough to explore the effects of different feedback agendas. For example, it might be interesting to explore whether students should be given feedback on

grammatical errors before discussing how well a sentence meets rhetorical goals, or vice versa.

There are two approaches to the problem of handling students' responses: surface pattern matching and parsing. The first approach, surface pattern matching, involves matching patterns or descriptions to a set of elements in computer memory. It was employed by Clancey (1979) in GUIDON, a system to tutor students in medical diagnosis. In GUIDON, a student may be asked, for example, to list the organisms causing an infection. The student may mistakenly omit organisms that the expert concluded were present, or add organisms the expert concluded were not present. The program treats the student's list as a pattern that must be matched to the organisms that the productions in the expert program would list; then hypothesizes which of the expert productions best account for the student's list. The tutor then uses the derived differential model as a basis for instruction.

Clancey's pattern matching approach can be made to work for sentences with a slight modification. Clancey indexes productions that list organisms directly from the organism names. However, with ordered elements such as words in a sentence, it is necessary to index with a type/token distinction. For example, the word 'fortitude' must be represented by as many tokens as there are alternative "expert" sentences in which the word 'fortitude' appears. To facilitate the matching process, the type/token links are bi-directional. In addition, any token must have links to the token immediately before it and immediately after it, as well as links to the productions which produced the "word" in that position.

Suppose the tutor is trying to match a student's response, "Humanity has hardly advanced in fortitude since that time, however, it has advanced in many other ways," to the list of possible sentence revisions itemized above. It will begin by trying to match the student's response to a list of tokens linked via the type/token link to the word 'Humanity.' By following the 'next' links, the match will continue without a problem until the word 'however,' is reached. When a mismatch is detected, the process attempts to account for the difference, much as a spelling checker does, as a deletion, addition or modification (such as progressive of "have"), subtracting points from the match according to a weighted distance measure. In addition, the mismatch can cause the process to pick up other possibilities.

The pattern matching approach is currently used in the PARNASSUS prototype. In this way, it is possible to arrive at the matches: (1) 'Humanity has hardly advanced in fortitude since that time, but it has advanced in many other ways,' and (2) 'Humanity has hardly advanced in fortitude since that time; however, it has advanced in many other ways.' Picking up both possibilities provides the tutor with potential for eventually working within the student's approach. The prototype has a spell checker and morphological component, so it is able to detect and take into account spelling errors and small grammatical mistakes. In the example above, the second sentence is currently rated the best match, reflecting a preference for a match to words over punctuation differences.

In order to achieve good performance, the features in the match eventually might have to be weighted according to the the tutor's assessment of what rules a

student knows. For example, when students first begin revision exercises, many have not mastered punctuation rules, so omission or insertion of punctuation should not count as heavily against a match as omission or insertion of words. Similarly, words involved in delsetions, additions, or transformations in the expert rules (e.g., although, but) should not count as heavily as words that are not so involved (e.g., humanity, advanced).

Pattern matching exploits strong expectations about a student's response and the capabilities of the student, but it treats the role of syntax in guiding the comprehension process superficially; that is, syntax plays a role in the expert module's condition/action rules for *producing* sentences, but there is no sharing of that knowledge with rules for *comprehending* sentences. The question naturally arises, shouldn't the system's understanding of the student's response be based on parsing techniques?

Parsing constitutes the second approach to handling students' responses. Parsing involves building a representation of a sentence. Parsers have been constructed that build syntactic representations, semantic representations, or both (Winograd, 1983). If parsing were used in PARNASSUS, the parsing process itself could detect problems with a student's response by registering mismatches between parsing expectations (e.g., an expectation that each sentence in a paragraph can be matched to a discourse schema) and a student's sentence. Parsing, however, would require (1) more attention to understanding the discourse expectations that are raised by different syntactic constructions than has been the case in semantic, expectation-driven parsers to date, and (2)

extending the techniques of robust parsing, that is, parsing ungrammatical input without break-downs in the process of building the representation. This would be an interesting research direction, but constitutes a research project in its own right. It is a direction taken by a considerable number of intelligent systems for language learning (Imlah & du Boulay, 1986).

As the scope of the PARNASSUS system increases, it remains to be seen whether the approach taken in the current system--matching to a set of alternatives--remains viable. Actual parsing of the students' responses may be necessary.

The Student Model

Decisions about the student's progress must be based on a "student model" that the program creates and maintains. When a student responds to a revision task, the response constitutes evidence for whether the student understands the principles required in order to perform the task successfully. If there were little evidence that the student understands the principles, the program ought to ask questions or pose exercises that examine the student's understanding. If there were evidence that the student did understand the principle, the program ought to shift the task to solving other problems. The PARNASSUS prototype currently does not *create* a model of the student, but it can *use* a model that is fed to it to direct the interaction. Informally, the model resembles the student model provided earlier in this paper as an example. The program must be expanded so that it creates and maintains the model.

Diagnosis. Consider again the example student response, "Humanity has hardly advanced in fortitude since that time, however, it has advanced in many other ways." Note that the response does not fix the coherence problem. Although it is clear that the student has only a partial solution, the program must be developed so that it can diagnose the cause of the student's failure to apply the relevant rhetorical principles. The failure could have several causes: the student may not have read the paragraph carefully, the student may be misled by the local coherence of the sentences, or the student may not know about the principle of emphasis. Deciding among multiple causes, known as the "apportionment of credit/blame" problem, concerns the tutor's ability to build and maintain an accurate student model. The tutorial interaction must be driven, in part, by the program's building such a model (Self, 1987).

Tutorial Strategies

The PARNASSUS prototype employs a tutorial strategy based on discovery learning. In the prototype, the tutor's response is based on the minimal tutorial intervention model developed in the WEST system (Burton & Brown, 1979): if there is a better revision, the program informs the student that it has a revision that it has rated better and asks the student whether he or she wants to try to produce it.

There are alternative, interesting tutorial strategies. First, the tutor could switch tasks, presenting the student with opportunities to discover principles (e.g., the principle of emphasis), perhaps through contrasting examples. Collins & Stevens

(1982) provide an excellent example of such a tutorial strategy. Second, the tutor could tell the student directly that a principle is violated (e.g., "Your sentence emphasizes 'the many ways humanity has advanced' and leads your readers to expect that the paragraph will continue to talk about it."). The tutor could model a careful reading of the paragraph or ask the student to do so (e.g., The tutor could model a "thinking-aloud" reading of the paragraph, telling the student the expectations that an expert reader might have after each sentence). A tutor which is based on learning by doing might, at this point, simply give the student its best revision. How successful each of these alternatives would be for teaching different students is an open research question. This is the sense in which a computer program can help us to find answers.

Tutorial agendas. The program must be able to accept partially correct responses, detect multiple problems, and order those problems for tutoring. Taking the example student response as a case in point ("Humanity has hardly advanced in fortitude since that time, however, it has advanced in many other ways."), the program must be able to accept this as a partially correct revision, detect its multiple problems (emphasis, grammar) and order those problems for tutoring. To achieve tutorial flexibility, the program must be able to set a tutorial agenda and deal with problems in the order in which they appear on the agenda. Again, altering such an agenda would alter the interaction, so the program could be used as a research tool. The prototype PARNASSUS program keeps a tutorial agenda that deals with emphasis problems first, provided that the student model indicates that the student has a problem with emphasis.

Problem-Solving Assistance in Context. The program must be able to provide problem-solving assistance in context. The program currently has no capability to help the student *while* the student is making a revision. Again, it is an open research question whether the best strategy is to intervene while the student is actually doing the revision or to wait until the student has completed a revision.

Estimates of Task Difficulty. Our work in designing heuristics to generate alternative sentences has given us more insight into task difficulty. For example, case studies of novices and experts indicate that some coherence problems are more difficult to detect and correct than others.[2] Work must be done to incorporate these insights into the design of the program's curriculum.

Why Build ITS?

In the introductory section, we saw the wide range of things that human tutors can do. The PARNASSUS project, like other work in ITS, addresses a small subset of the knowledge and processes that constitute intelligent tutoring. For example, the program cannot generate new problems for a student to work on, nor does it in any sense "comprehend" a paragraph or reason about the interaction or learn from the student.

Moreover, the program will not provide a complete curriculum for helping students learn to write effective sentences. Students' sentences can be inappropriate to their context for a wide range of reasons: problems with transition, coherence, conciseness, emphasis, old/new information, maintenance and signalling of focus, tone, and so forth. The program is currently limited to two

example tasks, each dealing with problems in coherence or emphasis. Furthermore, the project makes no claims that the examples will represent the most important skills students need to know, although care was exercised in picking an example for which there is evidence that the skills needed to do the task are skills that many students have not mastered.

As Yazdani (1987) observes, unlike CAI, ITS does not reflect a mature technology. Anderson et al. (1984) expect to spend ten person-years before having a complete curriculum for LISP and longer still before it will be reliable enough to be offered as a commercial product.[3] Work in ITS is not a panacea, nor will most of it be immediately practical. If does, however, offer not only the potential for advancing existing techniques for computer-assisted instruction in writing, but also for deepening our understanding of issues in its learning and teaching.

References

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge, Massachusetts: Harvard University Press.

Anderson, J. R., Boyle, C. F., Farrell, R., & Reiser, B. (July 1984). *Cognitive principles in the design of computer tutors* (Tech. Rep. ONR-84-1). Psychology Department, Carnegie Mellon University,

Bamberg, B. (Oct. 1984). Assessing coherence: A reanalysis of essays written for the National Assessment of Educational Progress, 1979-1979. *Research in the Teaching of English, 18*(3), 305-17.

Bereiter, C, & Scardamalia, M. (1987). *The Psychology of Written Compostion.* Hillsdale, NJ: Lawrence Earbaum Associates.

Brown, J. S., & Van Lehn, K. (1980). Repair theory: A generative theory of bugs in procedural skills. *Cognitive Science, 4*, 379-426.

Burton, R. R. (1979). An investigation of computer coaching for informal learning activities. *The International Journal of Man-Machine Studies, 11*, 5-14.

Carbonell, J. R. (1970). AI in CAI: An artificial intelligence approach to computer-assisted instruction. *IEEE Transactions on Man-Machine Systems, 11*, 190-202.

Charney, D. H., & Reder, L. M. (1986). Designing interactive tutorials for computer users. *Human-Computer Interaction, 2*(4), 297-317.

Clancey, W. J. (1979). *Transfer of rule-based expertise through a tutorial dialogue.* Doctoral dissertation, Stanford University,

Collins, A. M., Warnock, E. H., & Passafiume, J. J. (1975). Analysis and synthesis of tutorial dialogues. In G. H. Bower (Ed.), *The psychology of learning and motivation.* New York: Academic Press.

Collins, A., & Stevens, A. L. (1982). Goals and strategies for inquiry teachers. In R. Glaser (Ed.), *Advances in instructional psychology.* Hillsdale, NJ: Lawrence

Erlbaum Associates.

Collins, A., Brown, J. S., & Newman, S. E. (Jan 1987). *Cognitive apprenticeship: Teaching the craft of reading, writing, and mathematics.* (Tech. Rep. 403). Center for the Study of Reading, Urbana-Champaign,

Flower, L., Hayes, J. R., Carey, L., Schriver, K., & Stratman, J. (1986). Detection, diagnosis, and the strategies of revision. *College Composition and Communication, 37*(1), 16-35.

Freedman, S. (1979). How characteristics of student essays influence teachers' evaluations. *Journal of Educational Psychology, 71*(3), 328-338.

Goldstein, I. P. (1982). The genetic graph: A representation of the evolution of procedural knowledge. In D. Sleeman & J. S. Brown (Eds.), *Intellgent tutoring systems.* New York: Academic Press.

Halliday, M. A. K., & Hasan, R. (1976). *Cohesion in English.* London: Longman Group Ltd.

Hayes, J.R., & Flower, L. (1980). Identifying the organization of writing processes. In L. Gregg & E. Steinberg (Eds.), *Cognitive processes in writing.* Hillsdale, NJ: Lawrence Erbaum Associates.

Hayes, J. R., Flower, L., Schriver, K., Stratman, J., & Carey, L. (1985). *Cognitive processes in revision* (Tech. Rep. 12). Communications Design Center, Carnegie-Mellon University,

Imlah, W., & du Boulay, B. (1986). Robust natural language parsing in computer assisted language instruction. *System,* Vol. *13*(2).

Katona, G. (1940). *Organizing and memorizing: Studies in the psychology of learning and teaching.* New York: Hafner Publishing Co.

Koffman, E. B., & Blount, S. E. (1973). Artificial intelligence and automatic programming. *Proceedings of the Third International Joint Conference on*

Artificial Intelligence. Menlo Park, CA: SRI International Publications.

Laubsch, J. H. (1975). Some thoughts about representing knowledge in instructional systems. *Fourth International Joint Conference on Artificial Intelligence.* Tsibili, USSR.

Lewis, M. W., & Anderson, J. R. (1985). Discrimination of operator schemata in problem solving: Learning from examples. *Cognitive Psychology, 17,* 26-65.

Mann, W. C., & Moore, J. A. . (Jan-Mar 1981). Computer generation of multiparagraph English text. *American Journal of Computational Linguistics, 7*(1), 17-29.

McKeown, K. R. (1985). *Text generation.* Cambridge: Cambridge University Press.

Neuwirth, C. M., & Ogura, A. (1987). Implications of a theory of skill acquisition for the design of an ITS to teach effective written sentences. Paper presented at the Third International Conference on Artificial Intelligence and Education, Pittsburgh, PA, May 8-10.

Norman, D. A., Gentner, D. R., & Stevens, A. L. (1976). Comments on learning schemata and memory representation. In D. Klahr (Ed.), *Cognition and instruction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Ohlsson, S. (1987). Some principles of intelligent tutoring. In R. W. Lawlor & M. Yazdani (Eds.), *Artificial intelligence and education.* Norwood, NJ: Ablex.

O'Shea, T., & Self, J. (1983). *Learning and teaching with computers: Artificial intelligence in education.* Englewood Cliffs, NJ: Prentice-Hall.

Polya, G. (1957). *How to solve it.* New York: Dover.

Post, E. L. (1943). Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics, 65,* 197-263.

Schank, R. C. (1975). *Conceptual Information processing.* New York: North-Holland/American Elsevier.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals, and understanding: An*

inquiry into human knowledge structures. Hillsdale, NJ: Erlbaum.

Schank, R. C., & Riesbeck, C. K. (1981). *Inside computer understanding.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Self, J. A. (1987). The application of machine learning to student modelling. In R. W. Lawlor & M. Yazdani (Eds.), *Artificial intelligence and education.* Norwood, NJ: Ablex.

Sleeman, D., & Brown, J. S. (1982). *Intelligent tutoring systems.* New York: Academic Press.

Stevens, A., Collins, A., & Goldin, S. E. (1982). Misconceptions in student's understanding. In D. Sleeman & J. S. Brown (Eds.), *Intellgent tutoring systems.* New York: Academic Press.

Wenger, E. (1987). *Artificial intelligence and tutoring systems: Computational and cognitive approaches to the communication of knowledge.* Los Altos, CA: Morgan Kaufmann Publishers, Inc.

Wertheimer, M. (1959). *Productive thinking.* New York: Harper & Row. (rev. ed.).

Winograd, T. (1975). Frame representations and the declarative/procedural controversy. In D. Bobrow & A. Collins (Eds.), *Representation and understanding.* New York: Academic Press.

Winograd, T. (1983). *Language as a cognitive process.* Reading, MA: Addison-Wesley.

Wittrock, M. C. (1966). The learning-by-discovery hypothesis. In L. S. Shulman & E. R. Keislar (Eds.), *Learning by discovery: A critical appraisal.* Chicago, IL: Rand McNally & Company.

Woolf, B. P., & McDonald, D. D. (1984). Context-dependent transitions in tutoring discourse. *Proceedings of the National Conference on Artificial Intelligence.* Austin, TX.

Yazdani, M. (1987). Intelligent tutoring systems: An overview. In R. W. Lawlor & M. Yazdani (Eds.), *Artificial intelligence and education.* Norwood, NJ: Ablex.

Young, R. (1980). Arts, crafts, gifts, and knacks: Some disharmonies in the New Rhetoric. *Visible Language*, *14*(4), 341-350.

Acknowledgments

Preparation of this manuscript was supported, in part, by a grant from the Buhl Foundation.

Thanks to Ayami Ogura and Daniel Kahn for their work on the implementation of the prototype PARNASSUS system. Thanks to Davida Charney, David Evans, David Kaufer, Helen Schwartz, Erwin Steinberg, Richard Young, Gary Waller and the editors of this issue for their useful comments on earlier work and drafts.

Notes

1 The prototype is implemented in FranzLisp, a dialect of LISP, on a VAX/11-780.

2 Ayami Ogura, Topic-shift error detection and correction in written discourse. Unpublished manuscript, 1987.

3 A version of the LISP tutor is now beginning to be marketed as a product. See Anderson, J.R., Corbett, A.T. & Reiser, B. *Essential LISP.* Addison-Wesley Publishing, 1987, p. x.

PARNASSUS	Exercise 1.3
	Humanity has many miraculous qualities, but the most miraculous is its fortitude, its capacity to endure and to survive incredible hardships. In neolithic times, for instance, humanity endured adverse winters, ferocious animals, and grueling migrations.
	Humanity has hardly advanced in fortitude since that time. It has, however, advanced in many other ways.
	Indeed, some have argued that humanity's fortitude, at least in Western civilization, has declined; however, it is not humanity's capacity to endure that has declined, but rather the severity of the hardships Western humanity usually faces.

Figure 1. PARNASSUS Revision Exercise