MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AFIT/GST/ENS/86M-14

AD-A172 497

AN EXPERT SYSTEM FOR TUTORING

INTELLIGENCE ANALYSTS

THESIS

Richard O. Melvin
Major, USAF

AFIT/GST/ENS/86M-14

DTIC
ELECTE
S OCT 0 2 1986
E
D

Approved for public release: distribution unlimited

86 10 2 16 °

AN EXPERT SYSTEM FOR TUTORING
INTELLIGENCE ANALYSTS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Operations Research

Richard O. Melvin, B.S., M.B.A.

Major, USAF

June 1986

Approved for public release; distribution unlimited

Preface

The purpose of this research was to design and test a computer system to tutor signal analysts on radar concepts. Based on artificial intelligence techniques, this Intelligent Computer Aided Instruction (ICAI) system would assume some of the tasks of a human instructor. Although the design has been tested, much work remains before it can be used in an actual training environment. I hope this study will serve as the basis for additional research.

Without the help of several people, this endeavor would never have been completed. I would like to express my appreciation to Dave Hostler, Bill Kovacs, and Greg Stine from the Foreign Technology Division for suggesting and supporting this thesis. I thank Major Parnell, my faculty advisor, for his patient guidance during this project. I am also grateful to my readers, Maj Steve Cross and Capt Bob Hebert, who made numerous helpful suggestions. In addition to being a reader, Capt Hebert provided invaluable assistance with the computer system at the Artificial Intelligence Laboratory. Finally, I want to thank a group of special friends for their support while I was at AFIT.

Richard O. Melvin

ii

Table of Contents

Abstract

Intelligent Computer Aided Instruction (ICAI) allows a computer to perform some of the functions normally performed by a human instructor. This thesis describes the design and implementation of an ICAI system which presents radar principles to a student, tests him, finds out why he made an error, and then corrects the error.

To allow the system to be used for different subject domains, the knowledge required to teach was kept separate from the knowledge about the subject domain. During the design phase, the knowledge about teaching was partitioned into eight knowledge bases, the functions of these knowledge bases were described, and their interactions with one another were shown. During the implementation phase, each knowledge base was developed separately before being integrated into the system. Of these eight knowledge bases, the knowledge base which tests the student received the major emphasis. This knowledge base generates a multiple choice question, finds the answer, and creates plausible incorrect answers to serve as distractors. Although the radar range equation was used to test this knowledge base, this knowledge base performs the same functions with any equation which is entered into the system in its canonical form.

Although this thesis establishes the framework for an ICAI system and then demonstrates its feasibility, further

research is required before the system can be used in an actual training environment.

AN EXPERT SYSTEM FOR TUTORING INTELLIGENCE ANALYSTS

I. Introduction

Background

An organization, such as the Air Force, is faced with conflicting goals. Since it cannot meet them all, it must decide which goals are less important and sacrifice them, or it must find new ways of meeting them. An example of conflicting goals is the Air Force's need to expose future leaders to a variety of jobs, while at the same time insuring that its people have the expertise to accomplish the mission. This conflict is resolved through training programs. Whether this is formal training or on-the-job training (OJT), scarce resources of people, money, and time are used.

Since the Air Force wants to provide quality instruction, its OJT instructors are usually its most experienced people, i.e., the experts (Stuart et al, 1984:15). However, several disadvantages result from using experts as instructors. First, when the expert is instructing, he is not performing his primary job. Second, for the expert to be a good instructor requires that, in addition to being an expert at his primary job, he needs the skills to communicate the subject material to the student, diagnose why the student is having problems, decide how to
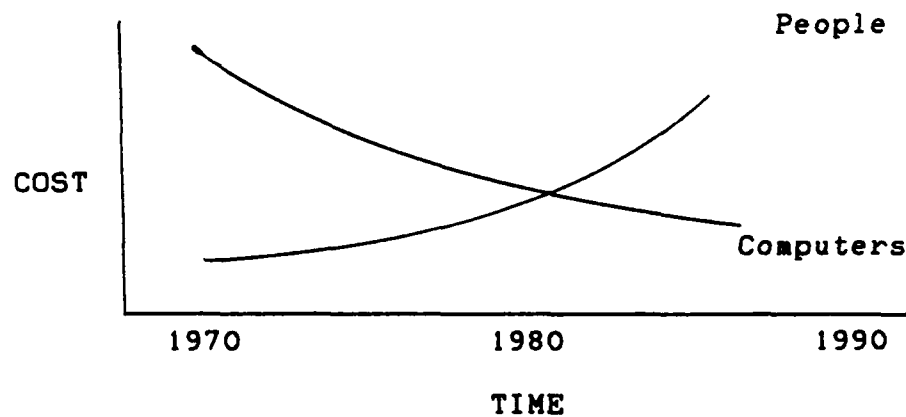
Figure 1.1   Cost of People Versus Computers

correct these problems, and then correct them.   Because of
the difference in the level of knowledge between the   expert
and   the   student,   the   expert may not understand   why   the
student   does   not   understand a concept   which   the   expert
thinks   is   obvious.   Assuming   the   expert   is   a   good
instructor, when he is leaves the organization, it no longer
benefits   from   his   teaching   skills.   Even   without   these
disadvantages,   there   a continuing demand for manpower   and
money to train people.

One   way to more effectively use experienced people   is
by   allowing   the computer to perform some of this   training
(Freedman and Rosenking,   1986:31).   As shown   in Figure 1.1,
the   cost   for personnel is increasing while   computers   are
becoming less expensive and more capable.   Although there is
an   initial   cost   for   the   computer   and   its   associated
software,   and   there is the continuing cost for maintenance
on   the system,   computer aided instruction   offers   several

I - 2

advantages. These benefits remedy the previous disadvantages by allowing experienced personnel to devote less time to training and more time to handling complex problems in their area of expertise, by consistently using proven methods for presenting material, and finally, by remaining in the organization indefinitely.

An organization which uses its experienced personnel as OJT instructors is the signal analysis branch of the Foreign Technology Division (FTD). When a person is assigned to FTD, he may have some background in electronics but little knowledge of radars and how to analyze radar signals. Since newly assigned personnel are trained locally by the experienced analysts, these analysts must divide their time between instructing and analyzing signals.

Compounding this problem is the relatively short time that a newly trained signal analyst remains with the organization before he is reassigned (Kovacs, 1985). The time for a person to begin training until he is qualified to analyze signals may take up to a year. However, once he has been fully trained, he will generally work as a signal analyst from between six months to two years before being reassigned. And then his replacement must be trained.

Problem Definition

Before a person can perform most Air Force jobs, he must possess some basic skills. However, the person providing this OJT training is usually the one with the most

experience; this is the person whose time the organization can least afford to lose. Since a person rarely remains in the same job longer than a few years and a large portion of this time may be spent in training, the use of the expert as an instructor causes a continuing drain on the organization's resources.

Thesis Objectives and Approach

The goal of this thesis is to develop a generic design for an intelligent computer aided instruction (ICAI) system capable of instructing, testing, and tutoring students on radar concepts. The design for the Signal Analyst Tutoring System (SATS) serves as the basis for further development. To permit maximum flexibility, the different functions of SATS are divided into individual knowledge bases. Prototype software is developed to test the feasibility of the design. Emphasis is given to different ways of representing knowledge to permit the computer to generate questions and answers for the material. This knowledge base is incorporated into the skeletal tutoring system and its capability is evaluated.

Scope

Due to the complexity of ICAI systems, most research focuses on only one aspect of a fully developed system (Barr and Feigenbaum, 1982:229). This is true for this effort. Although the system design is for a full tutoring system, many of the knowledge bases are developed only enough for

I - 4

the design to be evaluated. As a result, the level of development of the current prototype software does not allow it to be used for training.

Although SATS benefits from several capabilities offered by the M.1 expert system building tool, it also suffers from some of its limitations. Since the user interface is restricted to menus, the questions generated by the tutoring system is limited to a multiple choice format.

The purpose of this thesis is to develop ⌐ tutoring system rather than actual instructional material. Therefore, the lesson material used to test SATS comes from training materials (National Security Agency, 1981), textbooks (Skolnick, 1980; Stimson, 1983), and interviews (Stine, 1986). No attempt is made to use educational psychology for diagnosing and correcting the student's errors.

Assumptions

The primary assumptions involve the student. Since SATS is used to teach the more important concepts, it is assumed the student has read the assigned material about radars before using SATS. When answering questions, the student is expected to answer the questions to the best of his ability rather than making incorrect responses due to boredom, curiosity, or inattention.

The reader of this thesis is assumed to have a basic understanding of such artificial intelligence concepts as

knowledge bases and inference engines. The interested
reader can find additional information in the Handbook of
Artificial Intelligence, Harmon and King's Expert Systems,
and Waterman's A Guide to Expert Systems.

Equipment

SATS was developed on an IBM-AT in the Artificial
Intelligence Laboratory using version 2.0 of the M.1 expert
system building tool.

Overview

Chapter Two describes ICAI and recent research in this
area. The third chapter covers expert system building tools
and the different layers of knowledge in the tutoring
system. The fourth chapter concentrates on the different
knowledge bases and their interaction. Chapter Five
describes the prototype tutoring system developed for this
thesis. Chapter Six summarizes the research and makes
recommendations for further work.

II. Summary of Related Work

Introduction

Artificial intelligence allows computers to perform functions generally carried out by humans. These functions can be broken into natural language, robotics, and expert systems (Harmon and King, 1985:4-5). An expert system is "a computer program using expert knowledge to attain high levels of performance in a narrow problem area." (Waterman, 1986:11). The techniques used to build an expert system may be used to create programs which solve problems not requiring an expert; these are known as knowledge systems. As experience with knowledge systems grows, additional areas for application have also increased. One of these areas is intelligent computer aided instruction (ICAI) systems. This chapter introduces ICAI systems by looking at the three components into which an ICAI system may be divided and then describing some representative ICAI systems. These systems vary in approach, complexity, and subject area.

Components of an ICAI System

The goal of an ICAI system is to assume some of the duties of an instructor. ICAI systems can be broken into three components: problem-solving expertise, student model, and tutoring strategies (Barr and Feigenbaum, 1982:229). Current research in ICAI systems usually addresses only one of these areas.

The expertise forms the basis of student instruction. The form of expertise used depends on the subject area as well as the type of instruction to be accomplished. If the goal is to teach the student to perform a task properly, the task is often solved by an expert system or a simulator, and the student's answer is compared to that of the expert system. Another type of expertise is to simply present information. The form of the information must allow the tutoring system to reason over the information to generate questions and answers.

The student model contains "any information which a teaching program has which is specific to the particular student being taught" (O'Shea and Self, 1983:143). Two types of student models are the overlay model and the perturbation model (Sleeman and Brown, 1982:4). The overlay model, sometimes called a differential model, assumes that the student's knowledge is correct but is a subset of the expert's knowledge; the emphasis is placed on how they differ. The perturbation model represents the student's knowledge as not only less than the expert's knowledge but with mis-learned subskills.

The tutoring module determines how to teach the student. Ideally, the tutoring module considers both the student model and the instructional objectives. Some of the tutoring module's duties include what material to present, at what level to present it, and when and how to correct the student. One teaching style uses the Socratic method of

asking the student questions and letting him learn from his answers. Another teaching style, coaching, is accomplished by watching the student perform some activity and correcting him when he makes errors. If, however, the expert system solves the problem using methods other than those used by a human, the student may need to be tutored by a second less efficient but more articulate expert (Burton and Brown, 1982:82).

Generative Computer Aided Instruction

When the goal of ICAI is to provide information rather than teach a skill, a generative form of ICAI is used. In these systems, knowledge about the subject domain must be represented in a form that allows the system to reason over it and then generate questions and answers.

SCHOLAR. One such system, SCHOLAR (Carbonell, 1970; Gable and Page, 1980:265-267) teaches a student about the geography of South America. The knowledge about South American geography is represented in a semantic network. Facts, such as the location, size, climate, etc. of a country, form the nodes of the network and their relationships form the arcs between the nodes. An example of such a network is shown in Figure 2.1. SCHOLAR also uses the semantic network as a simple student model; it assumes that the student knows the material until proven otherwise. The system uses the Socratic method of teaching by asking the student questions. It also answers questions which the

South America

country-of   country-of

Brazil   Argentina

language   capital   language   capital

Portuguese   Brasilia   Spanish   Buenos Aires
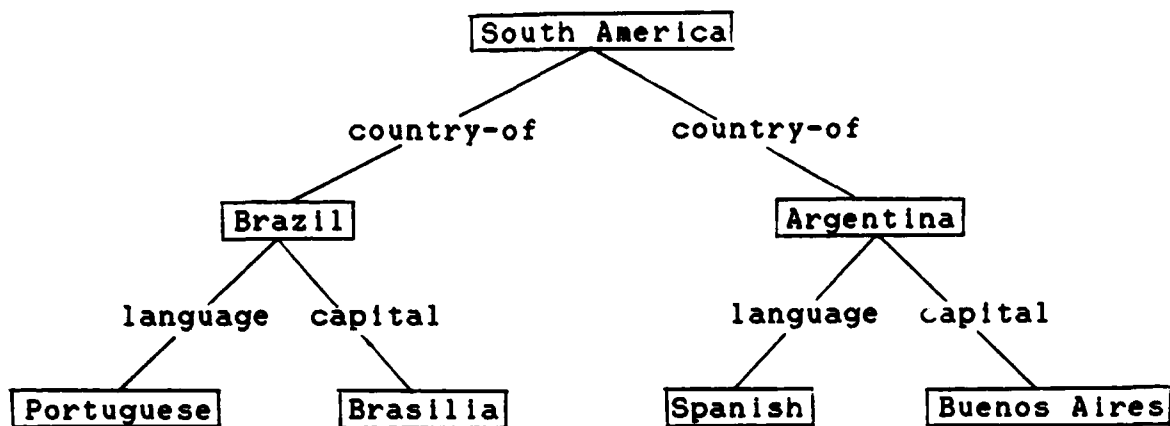
Figure 2.1   Semantic Network for SCHOLAR

student asks about South American geography. Since the
teaching knowledge and the subject knowledge are separate,
it is relatively easy to change the subject domain from
South American geography to African geography by changing·
part of the semantic network. A more substantive change,
such as changing the subject domain from South American
geography to anatomy, requires more extensive changes to the
semantic network. Even in this case, the teaching knowledge
in the program remains relatively stable. (Carbonell,
1970:191). This stability was demonstrated by building WHY,
a similar system, which teaches students about the causes of
rain (Stevens et al, 1982; Roberts and Ok-choon, 1983:8).

Task-Oriented Computer Aided Instruction

While SCHOLAR uses the generative form, most tutoring
systems teach a student a skill by letting him perform a
task, monitoring his actions, and then correcting his

errors.   This model of teaching behavior is known as task-oriented ICAI.

BUGGY. It is much easier for a tutor, whether a person or a computer, to detect a student error than to diagnose the cause of error.   Brown and Burton designed BUGGY to diagnose systematic errors using place-value subtraction as the subject domain.   After breaking subtraction skills into subskills, the researchers found ways these subskills could be applied to get the wrong answer;   eventually, 110 primitive bugs were found.   By inserting each bug into the correct procedure until the system's solution matches that of the student, a possible cause for the error can be found. However, the diagnosis is complicated when more than one type of bug can cause the same wrong answer, when several primitive bugs form a compound bug, or when the student's behavior is inconsistent (Burton,   1982:162,164).   Proper design of test problems is critical in detecting compound bugs.   One such test distinguishes among 1200 compound bugs with only twelve problems (Burton, 1982:172).   The program is interactive, giving the student additional problems when a student error is identified.   Once the diagnostic tests are complete, the diagnosis forms a student model; it can predict not only which future problems he will answer incorrectly but also his answers (Burton, 1982:160). BUGGY has also been used to train teachers to diagnose errors by solving problems with various bugs and requiring the teachers to detect the bug (Barr and Feigenbaum, 1982:281).

II - 5

MACSYMA Advisor. Whereas BUGGY finds the reason a student makes an error by evaluating only answers, more complicated problems require more information. The steps which the student takes in arriving at his answer are also needed. The MACSYMA Advisor attempts to solve this problem.

MACSYMA is an interactive computer program which helps scientists, mathematicians, and engineers solve mathematical problems (Genesereth, 1982:138). Because of its size and versatility, some users are unfamiliar with many of its features. The MACSYMA Advisor was developed to aid these users. The expertise in the MACSYMA Advisor is separate from MACSYMA; it has its own knowledge base for deducing the problem solving approach which the student is using, finding which part is incorrect, discovering why it is wrong, and then recommending the correct approach.

Although a particular procedure may be correct, it may be used at the wrong time or may not be appropriate for the problem. For instance, if the student is to solve the problem $X = 2 + 3 * 4$ he may perform the addition before the multiplication and decide that $X = 20$. Even though he added and multiplied correctly, he did it in the wrong sequence. By starting with the user's overall goal and the sequence the student uses to reach his goal, the Advisor tries to find which planning methods would have resulted in the student making the inputs he did. For example, if the overall goal is to find the root of a quadratic equation, the Advisor determines the different ways of accomplishing

this (e.g., finding the root by the quadratic formula or finding the root through factoring). These methods continue to be broken into sub-problems until they reach the point where the student needs to make an input. The Advisor compares the inputs required for each of these methods with the inputs actually made by the student to deduce which method he used. If there are several plans which the student may have used, the Advisor asks the student about the mathematical principles necessary for each plan. From the student's answers, the Advisor determines which principle the student needs tutoring. Rather than using the Socratic method for correcting the student's error, the Advisor simply tells him where he is wrong and suggests the proper operation. Although several approaches may be used to solve a particular problem, the Advisor recommends the approach that most closely follows the student's original approach.

SOPHIE. The previously described tutoring systems instruct rather than encourage the student to try out his own ideas. SOPHIE, an ICAI system designed to teach students techniques for electronic-troubleshooting, provides an experiential learning environment where the student can learn from his mistakes (Brown, 1982:229). Although SOPHIE has evolved through three systems, this discussion focuses on SOPHIE III.

While SOPHIE I and II use the general-purpose electronic simulator SPICE to simulate a faulted circuit,

II - 7

SOPHIE III allows the student to make measurements on an actual circuit. SOPHIE III consists of an electronics expert which reasons about electrical circuits, a troubleshooter which is concerned with measurements made on the circuit, and a coach which decides when to instruct the student. To give the electronics expert maximum generality, its designers used large amounts of general electronics information in its knowledge base; it uses this information to deduce the voltages and currents from measurements made by the student. The general electronics information is supplemented with circuit-specific information to inform the system about the circuit. During a consultation, the student tells SOPHIE the results of a measurement. As the student makes more measurements, the electronics expert uses the general electronics laws to find the voltages and currents experienced throughout the circuit for the measurements to be true. As the electronics expert continues this process, it eventually finds a contradiction between what is measured in the faulted circuit and what should be measured on its model of a good circuit. The troubleshooter then uses this contradiction to narrow the choice of possible faults. Since a major part of skillful troubleshooting is knowing which measurement to make, the troubleshooter evaluates the quality of the decision as to how efficiently it reduces the number of components suspected of being bad. The troubleshooter uses this criterion to evaluate the quality of a student's choice. It

also uses this criterion to suggest which measurement should be done next but only if asked to do so by the student. In addition to monitoring and commenting on the student's actions, the troubleshooter can explain its actions to the student as it troubleshoots a circuit. Finally, the coach decides when to instruct the student. For instance, if the student takes measurements which can be found from earlier measurements, it asks him some questions to see if the student recognizes this redundancy.

GUIDON. Since an expert system performs a task well, a student should gain experience by observing it operate. However, it was discovered that the structure of these expert systems do not permit them to effectively instruct. GUIDON is an ICAI system which uses the knowledge from the expert system MYCIN to teach medical students.

MYCIN is a medical expert system which provides advice about antimicrobial therapy for bacteremia and meningitis (Harmon and King, 1985:15). It acts like a consultant to a physician by asking him questions about a patient, diagnosing the patient's problem, and prescribing drugs to combat this problem. Like many expert systems, MYCIN is a production system based on a series of rules; the IF clauses, i.e., propositions, of the rule specify when the rule should be applied, and the THEN fact indicates the fact that is true if the rule is applied (Rich, 1983:31). Shown below is a sample production rule from MYCIN (Clancey, 1982:203).

```
IF (1) the gram stain of the organism
        is gram negative, and
    (2) the morphology of the organism
        is rod, and
    (3) the aerobicity of the organism
        is anaerobic,
THEN    there is suggestive evidence
        (0.6) that the genus of the
        organism is Bacteroides.
```

It was felt that, because of the expert knowledge in this system, it could be used to train medical students to diagnose and treat bacteremia and meningitis by presenting cases already solved by MYCIN. However, Clancey found that, although the production rules used in MYCIN can inform the user why the system does something, much of the information used by the expert system is implicit rather than explicit. That is, an experienced person can understand why the system does something while a student does not. GUIDON was developed to overcome these limitations by actively determining what the student really knows, presenting the information in an organized manner, and then explaining the expert's reasoning (Clancey, 1982:202).

GUIDON is a "case-method" tutor which leads a student through a case already solved by MYCIN. When MYCIN originally solves a case, it forms a tree of the goals and rules used to draw conclusions. This tree is used by GUIDON to find what conclusions MYCIN can draw based on the information available to the student; this is updated as the student receives more information. Using this tree as well as information on the student's history and competence,

GUIDON decides which of the conclusions formed by MYCIN should be known by the student.

Both discourse procedures and tutoring rules are used during the tutoring session. Discourse procedures tell GUIDON what it should do under certain circumstances. For instance, GUIDON starts a session using the discourse procedure CASE-DISCUSSION which first selects a case and then gives the student some preliminary information. GUIDON uses tutoring rules to select discourse procedures, choose domain knowledge, and update the student model (Clancey, 1982:210). An example of a tutoring rule used to update the student model is shown as follows (Clancey, 1982:220).

```
IF    (1) The hypothesis does include values that
          can be concluded by this domain rule, as
          well as others, and
      (2) The hypothesis does not include values
          that can only be concluded by this domain
          rule, and
      (3) Values concluded by the domain rule are
          missing in the hypothesis
THEN      Define the belief that the domain rule
          was considered to be -0.70.
```

Summary

Although interest in computer aided instruction has existed for several decades, most of the advances in ICAI systems have occurred within the last ten years because of increased processing speeds and reduced costs for computers. The goal of an ICAI may be to teach the student how to properly perform a task (task-oriented), or it's goal may be to present him with information about the subject domain (generative). The next chapter gives an overview of SATS by

describing the types of knowledge in the system and how the knowledge is partitioned into layers.

III. Conceptual Design

Introduction

When an individual starts working as a signal analyst, he may be unfamiliar with radar principles. SATS is designed to remedy this by tutoring these individuals. The design of SATS focuses on separating the knowledge about teaching from the knowledge about the subject area. The more these knowledge sources are separated, the easier it is to use SATS to tutor in other subject areas. This chapter shows how the knowledge comprising SATS is separated into layers and then describes the knowledge in each layer.

Hierarchy of Knowledge

Figure 3.1 shows the layers which form the hierarchy of knowledge for SATS as a software pyramid. Ascending this pyramid isolates the developer from the particular computer used and allows concentration on higher-level knowledge. However, when one selects the layer on which to develop a knowledge-based system, he must consider the tradeoff between the amount of time required to develop the system and its adaptability to meet different teaching requirements.

Although layers are isolated from one another, the lower layers constrain the form of the higher layers. Since starting near the bottom of the pyramid requires that all the higher layers be developed, the developer is free to design these higher layers to meet the requirements of the
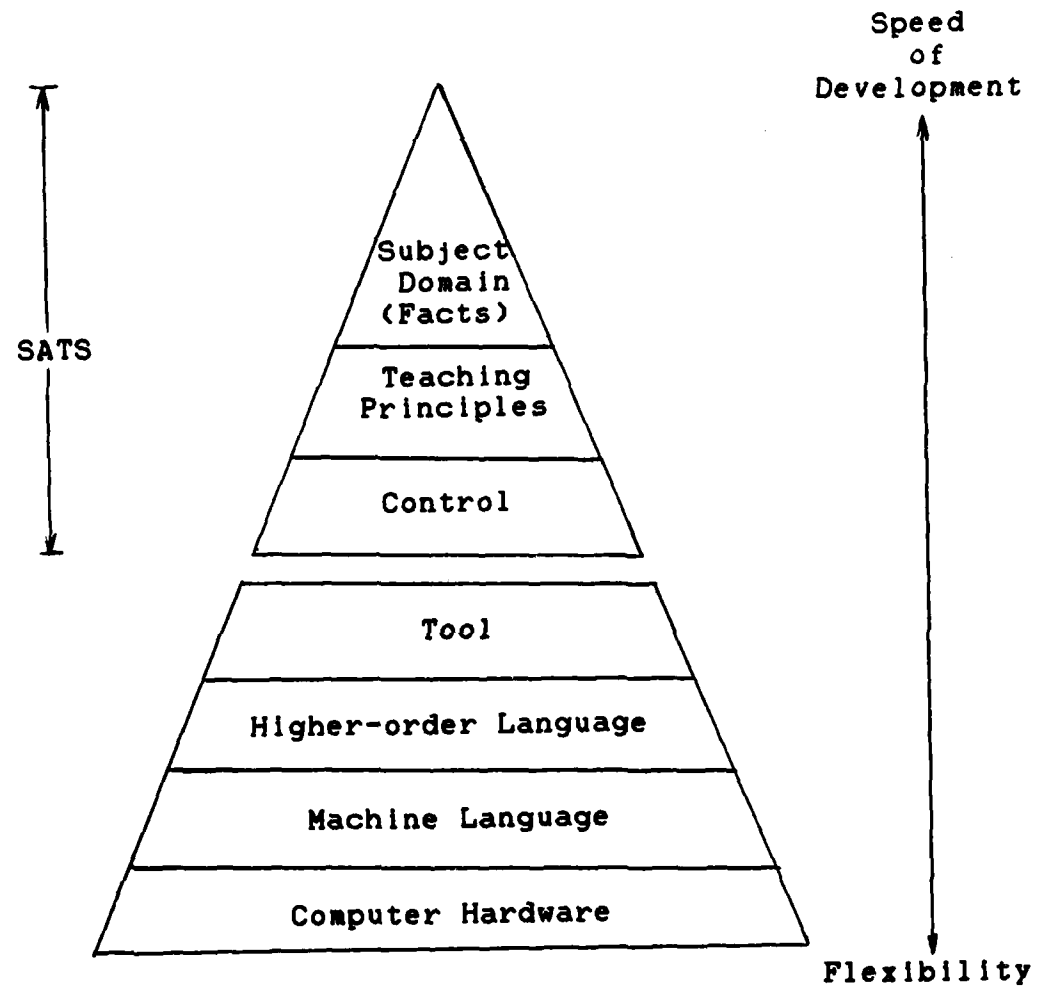
Figure 3.1   Software Pyramid

system. Thus, flexibility is the primary advantage of starting near the bottom of the pyramid. However, if a system has already been developed which meets most of the needs of the new system, a great deal of development time can be saved by adapting the original system to meet the needs of the new system. SATS starts at the expert system building tool layer and adds three more layers to the pyramid, thereby offering significant time savings to future
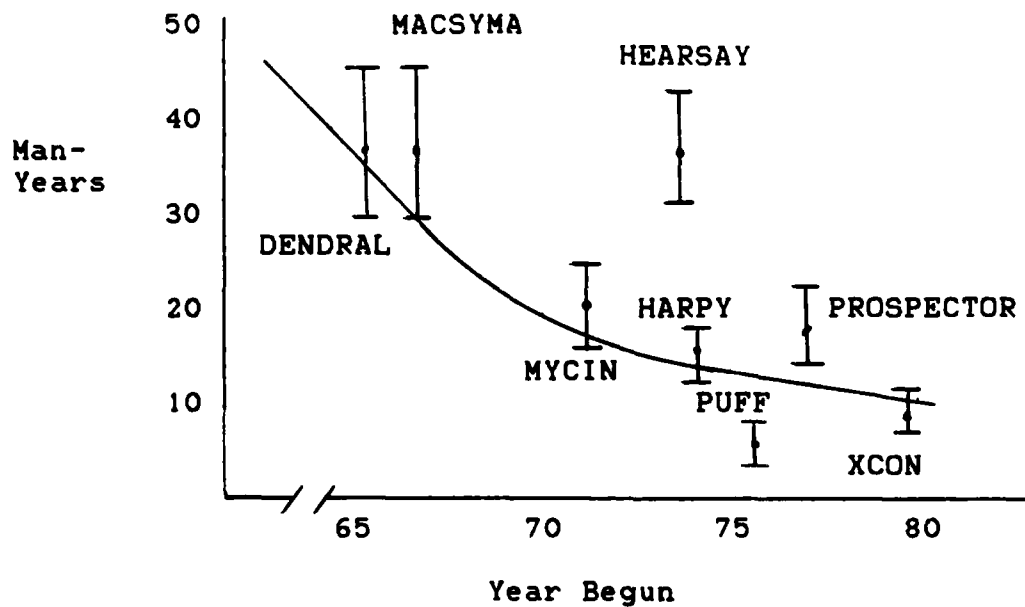
Figure 3.2   Time Required to Develop Expert Systems
(Davis, 1982:10)

tutoring system developers.   The effort required to   change

the   facts   about the specific subject domain is   relatively

minor,   but   the knowledge representation strategy   at   this

level is fixed.

Expert System Tools

"Expert   system   tools   are   programming   systems   that

simplify   the   job   of   constructing   an   expert   system"

(Waterman,   1985:80).   For   a knowledge-based system to   be

useful,   it needs an inference engine,   user interface,   and

explanation   capability.   As   shown   in   Figure   3.2,

approximately   ten   years   were   required   to   encode   the

knowledge   and create these features for the medical   expert

system MYCIN.   Later, the medical knowledge about meningitis

III - 3

and bacteremia was removed from MYCIN to create the expert system tool EMYCIN. By inserting medical knowledge about respiratory diseases into EMYCIN, the expert system PUFF was created in approximately five years. In exchange for the reduced time needed to prototype a new expert system, the developer must either insure that the problem solving paradigm of the new system closely matches the capabilities of the expert system tool or accept the constraints resulting from the tool.

Selecting an Environment

Several factors were considered in selecting the programming environment for SATS. Since this thesis emphasizes rapid design and evaluation of a tutoring system, a tool rather than a higher-order language was chosen. This allowed the effort to be concentrated on the problem rather than on construction of an inference engine, user interface, and explanation capability. Several other requirements further narrowed the choice of tools for the project: (a) after development at AFIT, SATS needed to be transported to FTD, (b) the tool should permit SATS to be maintained by someone without an extensive computer background, (c) the tool should offer features to help the knowledge engineer during system development, (d) the tutoring system should operate fairly quickly to minimize the amount of time a student spends waiting for the computer to decide what to do next.

Of the tools available, the M.1 expert system building tool comes closest to meeting these requirements. The person maintaining the system can learn the basic features of M.1 in a relatively short time through reading its documentation, examining sample knowledge systems, and attending a five-day course on M.1 offered as part of the professional continuing education course at the AFIT Artificial Intelligence Laboratory. He can then maintain and further develop the system. M.1 is easily transported. Once a knowledge system has been developed, it can be copied onto a floppy disk and later installed on another IBM-PC. Since many Air Force organizations have an IBM-PC (or compatible) computer, support should not be a problem. M.1 offers a relatively sophisticated inference engine and user interface. Several aids are also available to the knowledge engineer, including an explanation capability and other debugging devices. The one requirement in which M.1 is marginal is speed of operation. The knowledge engineer can improve its operating speed by properly constructing rules.

Control Rules

As shown in Figure 3.1, control rules form the layer between the expert system building tool and the teaching principles. These control rules govern the order and the way the teaching knowledge bases interact independently of the subject to be presented. This interaction, as shown in Figure 3.3, involves presenting material, asking

```
     ┌─────────┐
  ┌─→│ Present │
  │  │Material │
  │  └────┬────┘
  │       ↓
  │  ┌─────────┐
  │  │  Quiz   │
  │  │ Student │
  │  └────┬────┘
  │       ↓
  │  ┌─────────┐
  │  │Diagnose │
  │  │Problem  │
  │  └────┬────┘
  │       ↓
  │  ┌─────────┐
  │  │ Tutor   │
  │  │ Error   │
  │  └────┬────┘
  └───────┘
```
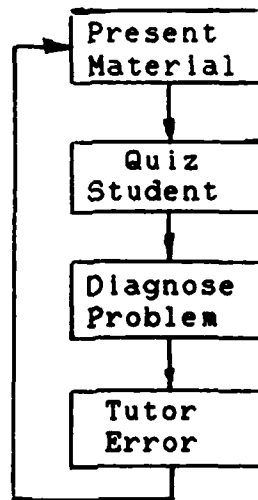
Figure 3.3 Teaching Process

the student questions and evaluating his answers, diagnosing
why he misses the questions, correcting his misconception,
and then repeating the process. Although modifying the
control rules affects the layers higher in the pyramid,
these effects are minimized because the rules in the upper
layers are separate from the control rules.

Teaching Principles

The knowledge comprising teaching principles is
partitioned into eight separate knowledge bases. These
knowledge bases accomplish specific teaching functions such
as testing the student, diagnosing the cause of his error,
and then correcting the problem. The operations of the
individual knowledge bases are discussed in Chapter Four.
Dividing the teaching functions into these knowledge bases
allows each one to be developed and tested independently

before being integrated into the full system. This also allows an individual knowledge base to be modified without affecting the other knowledge bases as long as the form of the information entering and coming out of the knowledge base remains the same. Attention can then be focused where it is needed - on the knowledge bases most affected by the modification. The effort required to modify SATS depends on the type of change made and the number of knowledge bases affected.

Facts

A list of facts comprises the knowledge about a specific subject domain. Since this is at the top of the pyramid and modifying it does not affect the lower layers, it can be easily changed. However, the knowledge representation strategy is constrained by the lower layers in the pyramid. To change the system at this layer basically involves substituting new facts for old facts in the same knowledge representation strategy.

Summary

A computer system should be based on several layers. In selecting the layer on the pyramid at which to start developing the system, the developer must consider the tradeoff between speed of development and design flexibility. By starting higher in the pyramid, the system can be developed more quickly, but constraints are placed on what the system can do since many of the features and

decisions are already made. SATS adds three additional layers to the hierarchical pyramid and can potentially save a developer a great deal of time in developing a new tutoring system. Chapter Four describes the system design for SATS by identifying the individual knowledge bases within these layers, discussing the type of knowledge in these knowledge bases, and then showing how they interact.

IV. <u>System Design</u>

<u>Introduction</u>

SATS' architecture allows the knowledge in SATS to be partitioned into layers and the knowledge in a particular layer to be further divided into knowledge bases. SATS is also designed to follow the same tutoring process used by an instructor. While the previous chapter discussed how the knowledge is split into layers, this chapter describes the tutoring process used by SATS; it examines the functions of the individual knowledge bases and then notes their interaction with one another. After describing SATS' current design, the chapter shows how SATS may be modified to perform task-oriented CAI rather than generative CAI. While this chapter addresses the design of SATS, Chapter Five discusses SATS as it is now implemented.

<u>Instruction - an Expert System Application</u>

Knowledge-based systems can be classified into categories according to the type of problem they solve. One of these categories, instruction, is a composite of three other categories: diagnosing, debugging, and repairing student behavior (Waterman, 1986:32-33). To be effective, each of these relies on the student model to reflect an accurate representation of the student's level of knowledge and his past performance.

<u>Diagnosis</u>. A general definition of diagnosis is "the process of fault-finding in a system . . . based on

interpretation of potentially noisy data" (Stefik et al, 1982:137). A diagnosis is made by assessing how the subject deviates from some standard. Difficulties in making a correct diagnosis include working with intermittent faults, compound faults, and inaccessible data. As applied to tutoring, an intermittent fault arises from inconsistent student behavior resulting from boredom or inattention. A compound fault occurs when the student makes several errors in arriving at his answer. Since the actual reason a student makes an error is inaccessible to the tutoring system, it must diagnose the probable cause based on symptoms.

Debugging. Knowledge-based systems in the debugging category "prescribe remedies for malfunctions" (Waterman, 1986:33). In the context of a tutoring system, debugging is the intermediate step between diagnosis and repair. While diagnosis finds out why the student makes an error, debugging decides how to correct it. In selecting a plan for correcting the error, the debugger must consider the plan's cost and availability. The plan should also accomodate the student's current knowledge and his learning style.

Repair. The last category involves knowledge-based systems which repair malfunctions by following a plan (Waterman, 1986:37). Using the plan created during the debugging phase, a tutoring system repairs the student's

```
Diagnosis ─────────► Debug
(Heuristics)         (Heuristics)
        ▲                     ╲
       ╱                       ╲
      ╱                         ▼
Error Category            Solution Category
      ▲                         │
      │                         │
      │                         ▼
Student Error             Specific Solution
  (Problem)
```
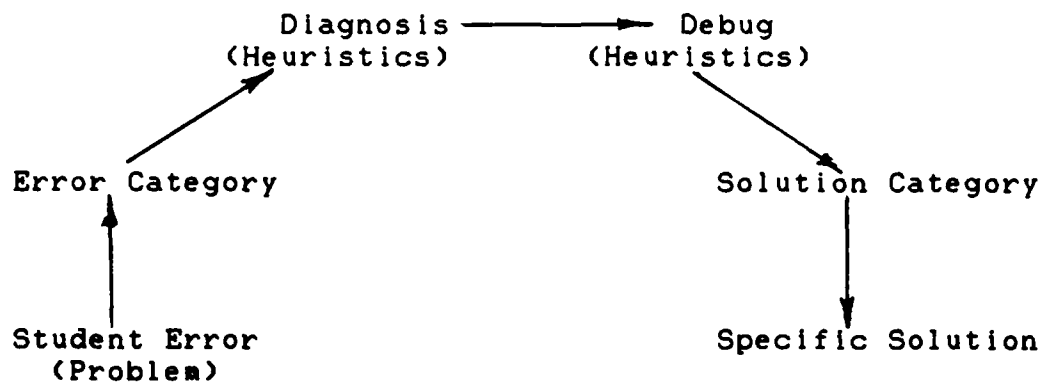
Figure 4.1 Heuristic Classification

behavior until it matches the standard. During this phase, care must be taken not to introduce additional student errors.

Heuristic Classification

Using these three instructional categories, there are several problem-solving techniques for determining the best way to correct the student's error. Simple classification merely uses one-to-one mapping between an error and its appropriate correction (Clancey, 1985:313). A knowledge-based system which uses heuristic classification more closely follows the problem-solving techniques used by experts (Clancey, 1985:291). In heuristic classification, a particular problem is placed into the appropriate error category, heuristics are applied to identify solutions for problems belonging to this particular category, and then the solution is applied to the particular problem. Figure 4.1 shows how this applies to a tutoring system using a specific

student error as the problem. For example, the student may not know that "c" represents the speed of light. This error is then placed into its appropriate error category. Using heuristics, or rules of thumb, a decision is made concerning the probable cause for the student making this type of error. Other heuristics are then applied to determine the best way to correct this particular type of error. Finally, this solution category is applied to the specific error and the student is given remedial tutoring.

SATS - System Design

Using the concepts discussed above, SATS leads a student through a series of lessons on radar principles by instructing a radar concept and then asking some questions. If he answers correctly, the system continues instructing. However, if he answers incorrectly, SATS attempts to determine why the student answered incorrectly and then provides remedial tutoring. After correcting the problem, SATS continues leading the student through the lesson.

SATS incorporates the three components of an ICAI system discussed in Chapter Two; these are the problem-solving expertise, the student model, and the tutoring strategies. In keeping with SATS' modular design, these are further divided into the eight knowledge bases (KB) shown in Figure 4.2. These knowledge bases contain the facts and rules used by an instructor. Included in the problem-solving expertise component are the Presentation-KB, the

SIGNAL ANALYST TUTORING SYSTEM (SATS)

COMPONENT	KNOWLEDGE BASE	FUNCTION
PROBLEM-SOLVING EXPERTISE:		
	LESSON	Contains a series of related concepts
	PRESENTATION	Leads the student through a lesson
	QUESTION GENERATOR AND SOLVER	Develops questions and answers over material
STUDENT MODEL:		
	STUDENT MODEL	Contains information about the student's knowledge
	QUIZZING	Questions the student to create and update the student model
TUTORING STRATEGIES:		
	DIAGNOSTICS	Determines why the student misses a question
	METATUTOR	Decides how to present information to the student
	TUTOR	Presents material to the student based on instructions from the Metatutor

Figure 4.2 Description of SATS

Lesson-KB, and the Question Generator and Solver-KB. The student model component consists of the Student Model-KB and the Quizzing-KB. The tutoring strategies component contains the Diagnostics-KB, the Metatutor-KB, and the Tutor-KB.

Control. When a student uses SATS, these knowledge bases interact with one another as shown in Figure 4.3. A series of control rules guides the tutoring session and governs knowledge base interactions. The current goal of SATS is teaching the student about radars. Initially, the Quizzing-KB questions the student about his background; this information forms the initial Student Model-KB. After ensuring the prerequisites are met, the appropriate lesson is selected from the Lesson-KB. Using this material, the Presentation-KB leads the student through the lesson. After the Presentation-KB instructs the student on a particular concept, the Question Generator and Solver-KB makes up appropriate questions. For each question generated, this knowledge base finds the correct answer as well as several reasonable incorrect answers to serve as distractors. These questions allow the student to apply what he has learned and gives SATS feedback on how well the student is learning the information. If the student answers correctly, the Presentation-KB presents the next concept. If, on the other hand, the student admits that he does not know the answer or if he answers incorrectly, the Diagnostics-KB determines the cause of the student's problem from information in the Student Model-KB; if more information is needed, the

```
              ┌─────────────────┐
              │   Quizzing KB   │
              │(Information for │
              │ Student Model)  │
              └─────────────────┘
                       │
                       ▼
              ┌─────────────────┐
              │ Student Model KB│
              │   (Initital)    │
              └─────────────────┘
                       │
                       ▼
              ┌─────────────────┐
              │   Lesson KB     │
              │ (Select Lesson) │
              └─────────────────┘
                       │
                       ▼
              ┌─────────────────┐
              │ Presentation KB │
              │ (Present Lesson)│
              └─────────────────┘
                       │
                       ▼
   Correct   ┌─────────────────┐   Incorrect
   Answer    │Question Generator│   Answer
             │ and Solver KB   │
             │ (Test Material) │
             └─────────────────┘
                                   Not Know
                                   Answer
   Lesson
   Complete                ┌─────────────────┐
                           │ Diagnostics KB  │
                           │(Reason Student  │
                           │ Missed Question)│
                           └─────────────────┘
                                    │
                                    ▼
                           ┌─────────────────┐
                           │   Quizzing KB   │
                           │(Additional Information│
                           │ on Student Knowledge)│
                           └─────────────────┘
                                    │
   ┌─────────────────┐     ┌─────────────────┐
   │  Metatutor KB   │     │ Student Model KB│
   │(Select Tutoring │     │   (Updated)     │
   │ Presentation)   │     └─────────────────┘
   └─────────────────┘
            │
            ▼
   ┌─────────────┐  ┌───────────┐  ┌─────────────────┐
   │  Tutor KB   │  │Quizzing KB│  │ Student Model KB│
   │ (Present    │→ │(Additional│→ │   (Updated)     │
   │ Material)   │  │Information│  └─────────────────┘
   └─────────────┘  │on Student │
            │       │Knowledge) │
            ▼       └───────────┘
   ┌─────────────────┐
   │Question Generator│
   │ and Solver KB   │
   │ (Confirm Student│
   │  Understands    │
   │    Error)       │
   └─────────────────┘
```

Figure 4.3   Interaction among Knowledge Bases

Quizzing-KB asks additional questions. Once the Diagnostics-KB determines the cause, the Quizzing-KB asks more questions to confirm it. The Metatutor-KB then decides the appropriate tutoring approach and directs the Tutor-KB to present the material. While the Tutor-KB is presenting the material, the Quizzing-KB asks the student questions; the Student Model-KB is updated based on the student's answers. After the remedial tutoring is finished, the Question Generator and Solver-KB confirms that the student's problem is corrected. SATS then proceeds with the lesson.

Knowledge Bases. Having considered the interaction of the knowledge bases from the viewpoint of the control structure, this same interaction is now examined from the viewpoint of the individual knowledge bases by explaining the function of the knowledge base and then showing how it interacts with the other knowledge bases.

Lesson Knowledge Base. The Lesson-KB contains a number of lessons. The student reads the assigned material before covering a lesson with SATS. Each lesson in SATS then emphasizes some of the more important concepts. Once the concept is mastered, the next concept is tutored.

Presentation Knowledge Base. The Presentation-KB leads the student through a particular lesson. After a concept has been explained, the student is asked a series of questions. Depending on how he answers the questions, the Presentation-KB either continues leading the student through the presentation or the Diagnostics-KB determines why the

student made an incorrect response before the lesson is continued.

Question Generator and Solver Knowledge Base. The Question Generator and Solver-KB develops questions based on the type of material covered and on the student's experience level. As the student answers these questions, both he and SATS learn how well he understands the material. From the results of these answers, SATS orients the session to meet the student's needs.

Student Model Knowledge Base. To tailor the teaching process to the student, SATS must know the student's level of knowledge and experience. The Quizzing-KB initially asks the student a series of questions from which the initial Student Model-KB is created. Throughout the session, the Student Model-KB is updated from the student's responses to questions developed by the Question Generator and Solver-KB and from questions asked by the Quizzing-KB. Other knowledge bases use the Student Model-KB to select the appropriate lesson to present to the student, to develop questions with the proper degree of difficulty, to diagnose the student's error, and to determine the correct type of tutoring presentation. At the end of a consultation, the Student Model-KB is retained.

Quizzing Knowledge Base. The Quizzing-KB is first used to build the initial Student Model-KB. It is also used to update the Student Model-KB if the Diagnostics-KB needs additional information before determining the student's

problem. Finally, SATS asks questions during the remedial tutoring process to update the Student Model-KB and allow the Tutor-KB to tailor its instruction to the student.

Diagnostics Knowledge Base. If, during a lesson, a student does not answer a question correctly, the Diagnostics-KB discovers the cause based on the student's error and the Student Model-KB. If the information in the Student Model-KB is insufficient for the Diagnostics-KB to determine the cause, the Quizzing-KB asks the student the appropriate questions. After this information becomes available, the Diagnostics-KB determines the cause and the Quizzing-KB confirms it.

Metatutor Knowledge Base. Once the Diagnostics-KB determines the cause, the Metatutor-KB decides how to present the remedial tutoring based on the Student Model-KB as well as on the particular problem. The Metatutor-KB operates through a series of metarules (that is, rules about rules) consisting of a premise that identifies when to apply the metarule and a series of steps to be accomplished if the premise is satisfied. For example, the premise may state that for a certain type of problem and for a student with a given experience level, the problem should be introduced, the concept defined, and then an example given. It then directs the Tutor-KB to follow these steps. However, in giving the example, a more basic concept may need to be explained; the Metatutor-KB then decides how to present this more basic concept before the original problem is corrected.

Tutor Knowledge Base. The Tutor-KB has several forms of explanation (for example, definition, analogy, etc.) for presenting the different concepts. It also has several levels for each of these forms of explanation. For example, the Tutor-KB has several explanations describing when to apply a certain principle, several definitions for a particular term, etc. Providing several choices for each of these forms of explanation gives SATS the flexibility to tailor the total presentation to the student's level of knowledge.

Task-Oriented Instruction

The goal of SATS may be changed from communicating general knowledge about a subject domain (generative CAI) to teaching a student how to perform a task (task-oriented CAI). This requires extensive changes to the problem-solving expertise component of the tutoring system. The Lesson-KB, Presentation-KB, and Question Generator and Solver-KB, which are included in this component of the current SATS, are replaced by a Task-KB and a Guiding-KB. These are described below. For this new system, the control rules remain the same, and the knowledge bases for the other components require only moderate changes.

Task Knowledge Base. Before a tutoring system can teach a student how to perform a task, it must be able to perform the task itself. This knowledge base is often an expert system.

<u>Guiding Knowledge Base</u>. Using the Task-KB as the source of expertise, the Guiding-KB leads the student through the problem-solving process. The goal of the system may be for the student to arrive at the same final answer as the Task-KB. In this case, the Task-KB solves the case completely before tutoring the student, the student acts as a consultant, and his final answer is compared with that of the Task-KB. However, to give the student more freedom, the Guiding-KB may direct the Task-KB to find intermediate solutions based on the information available to the student and then compares these answers with the student's answers.

Summary

SATS is designed to imitate the tutoring process used by an instructor. That is, when a student makes an error, SATS attempts to diagnose the cause of the error, decide how to correct it, and then correct it. Whereas ICAI systems are generally divided into the three main components of problem-solving expertise, student model, and tutoring strategies, SATS is further divided into eight knowledge bases to promote modularity. This modularity allows the goal for SATS to be altered from generative CAI to task-oriented CAI primarily by modifying the knowledge bases in the problem-solving expertise component. While this chapter explains how SATS will function once it is fully developed, the next chapter describes it as it now exists.

V. Implementation of SATS

Introduction

SATS' modular design, as described in the previous chapter, allows the individual knowledge bases to be developed to different levels and yet allows the integrated system to be tested. For this project, most of the developmental effort is directed towards the Question Generator and Solver-KB. After giving a brief overview of the current status of SATS' development, this chapter discusses how each knowledge base is implemented.

Overview of SATS

Figure 5.1 shows who performs the various functions required by SATS. It also indicates the degree of development for each knowledge base and the relative ease of applying a particular knowledge base to another subject domain. Since the inference engine and user interface are supplied by M.1, most of the effort can be directed towards developing the knowledge bases. The degree of development for the control structure allows all of the teaching knowledge bases to be used. While the basic structures for the teaching knowledge bases have been developed, they need to be maintained or extended by an instructor with some experience with M.1. Since most of the developmental effort centers around the Question Generator and Solver-KB, the extent to which the other knowledge bases are developed depends on the requirements of the Question Generator and

FUNCTION	PERFORMED BY	DEGREE OF DEVELOPMENT	ADAPTABILITY TO ANOTHER DOMAIN
Inference Engine	M.1		
User Interface	M.1		
Control Structure	KE	Advanced	High
Teaching Knowledge Base:			
Lesson	INS	Basic	Low
Presentation	INS	Basic	Low
Question Generator and Solver	INS	Advanced	High
Student Model	INS	Medium	Medium
Quizzing	INS	Basic	Medium
Diagnostics	INS	Basic	High
Metatutor	INS	Basic	High
Tutor	INS	Basic	High

Key: M.1 = Expert System Tool
      INS = Instructor Familiar with M.1
      KE = Knowledge Engineer

Figure 5.1 Function Matrix for SATS

Solver-KB. Except for the Lesson-KB and the Presentation-KB, whose knowledge is specific to a particular subject domain, the knowledge bases are relatively easy to adapt to a different subject domain.

Problem-Solving Expertise

As discussed in Chapter Two, the problem-solving expertise component of an ICAI contains the material for the student to learn. This expertise may be general knowledge about the subject domain (generative CAI), or it may teach the student how to perform a specific task (task-oriented CAI). As the purpose for SATS is to instruct general knowledge about radars, the generative type of ICAI forms the operating model of SATS.

Lesson Knowledge Base. The material used to teach the student is in the Lesson-KB. The four types of instructional material are facts, concepts, principles, and procedures (Wedman and Stefanich, 1984:23). A fact is a piece of information, such as R represents range, and has to be memorized. A concept contains a set of ideas or objects which have a common attribute; antenna gain is a concept. A principle shows how concepts are related; an example is the relationship between the range at which a target can be detected and the antenna gain of the radar. A procedure is a way to accomplish some task (for example, tuning a signal on an oscilloscope). SATS tutors and tests the first three types of instructional material. These are examined more

closely in a later section on the Question Generator and Solver-KB. The fourth type of instructional material, procedures, was discussed in the previous chapter (see task-oriented instruction).

For this project, the lesson material is based on the radar range equation. Although the radar range equation exists in several forms, the following equation is used for purposes of this discussion.

$$R = \left(\frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 P_r} \right)^{1/4} \tag{1}$$

where

R   is range
P_t is power transmitted
G is antenna gain
λ is wavelength
σ is radar cross section
P_r is power received

In this form, the radar range equation relates the effect that different parameters, such as antenna gain or the power transmitted, have on the range at which a target can be detected. The material in this knowledge base is easily revised by changing the text.

Presentation Knowledge Base. In SATS, the presentation of a lesson is controlled through an ordered listing of the concepts to be taught. Although this list fixes the way a lesson is developed, it can be easily changed by adding, deleting, or reordering concepts in the list.

```
        ┌─────────┐              ┌─────────┐
   ┬    │ Student │              │ Lesson  │
   │    │ Model   │              │ Area    │
        └─────────┘              └─────────┘
INPUTS         │                      │
   │           ▼                      ▼
   ┴    ┌──────────────────────────────┐
        │            Terms             │
        ├──────────────────────────────┤
        │         Definitions          │
        ├──────────────────────────────┤
   ┬    │          Equations           │
   │    └──────────────────────────────┘
OUTPUTS        │         │          │
   │           ▼         ▼          ▼
   ┴    ┌──────────┐ ┌────────┐ ┌────────────┐
        │ Question │ │ Answer │ │ Distractors│
        └──────────┘ └────────┘ └────────────┘
```
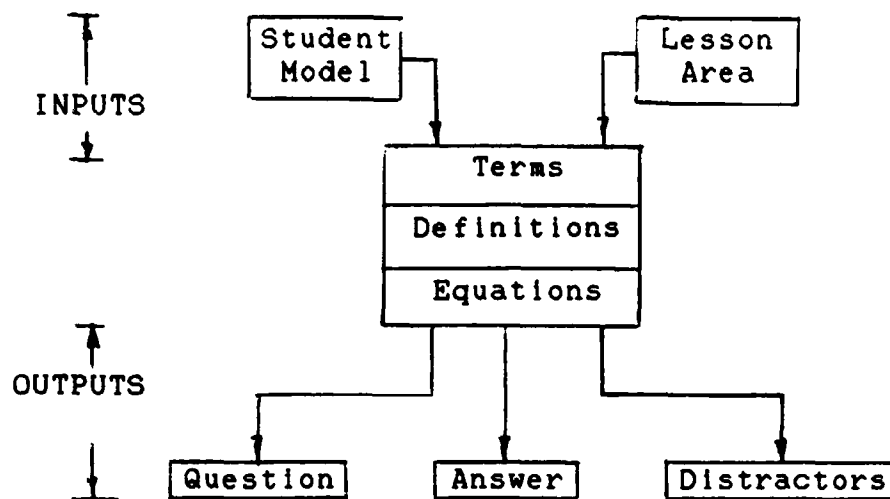
Figure 5.2 Question Generator and Solver Knowledge Base

Question Generator and Solver Knowledge Base.   This is
the most developed knowledge base.   It tests the student on
material   represented   by   the   first   three   types   of
instructional   material   mentioned   in   the   Lesson-KB   section.
Figure   5.2 shows the inputs and outputs for this   knowledge
base.   The   knowledge   representation strategy for each   of
tnese allows the Question Generator and Solver-KB to   reason
over   the   knowledge and generate   questions,   answers,   and
distractors.   These   distractors   are   answers   which   are
incorrect   even   though   they   appear   reasonable.   The
techniques   used in this knowledge base are described in the
following   subsections.

Facts.   One   of   the first things a   new   student
needs   to   learn   is   the terminology used   in   the   subject
domain.   Since   the   lesson   material   developed   for   this
project   focuses   on the radar   range   equation,   the   terms

tutored by SATS are those used in the equation. As with most equations, the terms in the radar range equation are represented by symbols. After the student is familiar with the symbols, the Question Generator and Solver-KB creates a question by randomly selecting one of these symbols and the term it represents. It then randomly selects several other terms used in the radar range equation as reasonable distractors. For a question of this type, SATS might ask the student what the symbol σ represents. In addition to the correct answer, the student's other choices could include wavelength, antenna gain, and power received.

Concepts. In addition to associating a symbol with the proper term, the student needs to understand the concept represented by the term. This type of material can be tested by having the student define the term (Wedman and Stefanich, 1984:24). Using the same methods described in the preceding paragraph, a term and its appropriate definition is selected. Definitions to the other terms in the radar range equation are randomly selected as distractors.

Principles. After becoming familiar with the basic terminology, the student needs to know how the terms in the radar range equation are related to one another. He needs to understand both the qualitative and the quantitative relationships. Qualitative reasoning looks at the gross features of a problem. For example, if a ball is released on an inclined plane, qualitative analysis predicts

that it will roll to the bottom whereas quantitative analysis determines the ball's velocity at the bottom of the inclined plane (de Kleer, 1979:14). Qualitative reasoning not only helps determine how to attack a problem but also provides a structure for quantitatively solving it (de Kleer, 1979:12,14).

For SATS to generate questions, answers, and distractors for the radar range equation, the equation needs to be entered as a binary list. Equation 1 appears below in its more general form

$$R = \left(\frac{P_t G^2 \lambda^2 \sigma}{(4\pi)^3 P_r} \right) 1/4 \qquad (1)$$

and then in its canonical form.

R = [root,[quotient,[product,[product,P_t,[exponent,G,2]],

[product,[exponent,λ,2],σ]],

[product,[exponent,[product,4,π],3],

P_r]]

4]

Initially, the student is concerned with only qualitative relationships. For instance, does increasing the antenna gain (G) increase, decrease, or have no effect on the detection range for a radar? To create a question and then find its correct answer, the Question Generator and Solver-KB randomly selects a term; for this example, assume antenna gain is selected. Then, by applying a series

of rules, it determines if antenna gain is in the equation and, if it is, whether it is in the first part (numerator) or last part (denominator) of the list. Since antenna gain is in the numerator, SATS knows that range is directly related to antenna gain. After SATS finds the answer, it tests the student on this relationship. Regardless of the qualitative relationship, the choices available to the student are that the relationship between terms is direct, inverse, or unrelated.

In addition to just knowing whether the relationship between antenna gain and detection range is direct, inverse, or unrelated, the student should appreciate the sensitivity between the two terms. Again, for the Question Generator and Solver-KB to find the answer, the equation must be in the canonical form shown earlier. After randomly selecting a term from the equation, the Question Generator and Solver-KB randomly selects an amount to vary it. For this example, assume that antenna gain (G) is doubled. To find the correct answer, the Question Generator and Solver-KB must find the qualitative relationship and the exponential relationship between range and antenna gain. Once it finds the answer, the Question Generator and Solver-KB creates distractors by varying the true qualitative and exponential relationships between range and antenna gain. A sample question is shown below.

How does increasing G by a factor of 2.0 affect R?

    1.   0.5
    2.   0.707
    3.   1.414
    4.   2.0

Student Model

The student model component contains information on the student's familiarity with the subject domain as well as his past performance in answering questions. In SATS, this component consists of the Student Model-KB and the Quizzing-KB.

Quizzing Knowledge Base. This knowledge base initially asks the student a series of questions about the subject domain; from these questions, an initial student model can be formed. Although the system design allows the Quizzing-KB to later modify the Student Model-KB, the present system only uses the Quizzing-KB to form the initial Student Model-KB.

Student Model Knowledge Base. The Student Model-KB influences how the tutoring system interacts with the student. From the questions asked by the Quizzing-KB, the student's knowledge about different parts of the lesson is classified as basic, average, or advanced. This allows the tutoring to better focus on his needs. Although Figure 5.3 shows a student with an advanced understanding of terms and definitions and a basic understanding of the qualitative and quantitative relationships between terms, this is only one of eighty-one possible student models in the present

Types of Lesson Material

	Termi-nology	Defini-tions	Quali-tative	Quanti-tative
Basic			X	X
Average				
Advanced	X	X		

Student Experience (label on left spanning Basic/Average/Advanced rows)

Figure 5.3 Example Student Model

system.   Figure   5.4 shows that although an initial student model is created based on questions from the Quizzing-KB, it is updated based on the student's answers to test questions. The student's classification in a particular area may remain the   same   or   change to a higher   or   lower   classification level.   This   classification   level affects the   number   of questions   he is asked and the type of remedial tutoring   he receives after incorrectly answering a question.   To update the student model after the student has answered a series of questions, the following production rule is used.

```
IF    the number of questions answered correctly is X and
      the number of questions asked is Y and
      X/Y > .5 and
      X/Y < .75
THEN  the student's proficiency level is average.
```

Tutoring Strategies

The   tutoring strategies component determines   why   the student   makes   an   error,   how   to   correct   it,   and   then corrects it.   The Diagnostics-KB, the Metatutor-KB, and the Tutor-KB perform these functions.
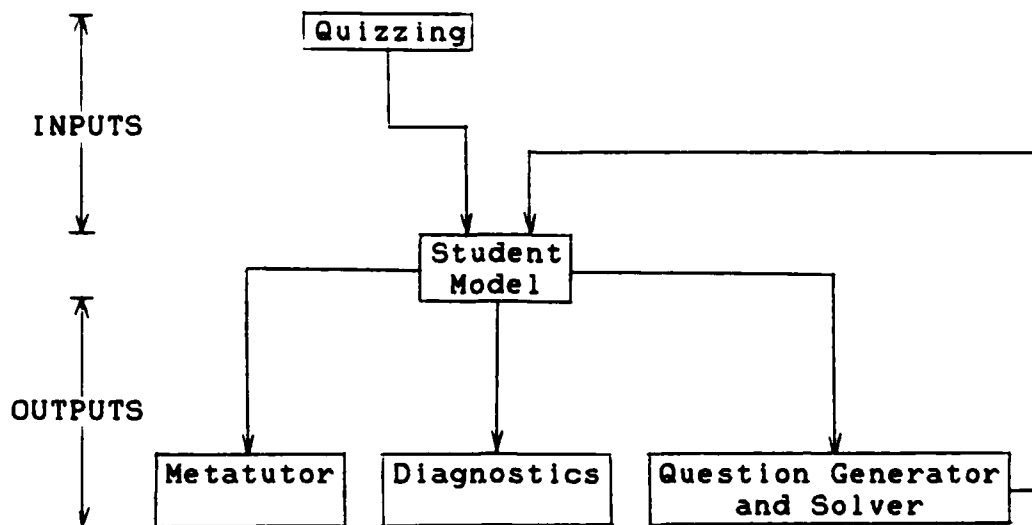
Figure 5.4   Student Model Knowledge Base

Diagnostics Knowledge Base.   No attempt is made to
include diagnostic principles from cognitive psychology in
this system;   that is, the Diagnostics-KB only looks at what
the student misses rather than why he misses it.   Since SATS
is restricted to asking questions with multiple choice
answers,   the   distractors are generated based on possible
errors the student may commit.   Therefore, if the student
selects an incorrect answer, the Diagnostics-KB assumes that
the   student is using the same incorrect procedure employed
by the Question Generator and Solver-KB to generate the
distractor.

Metatutor Knowledge Base.   This knowledge base uses
information from the Diagnostics-KB to decide what to tutor,
utilizes information from the Student Model-KB to determine
the   depth of tutoring, and combines these to decide how to

correct the student's error. For instance if a student misses the effect of doubling the antenna gain on the detection range, he receives remedial tutoring based on his experience level and on the incorrect answer he selects.

Tutor Knowledge Base. This knowledge base provides the actual remedial tutoring based on information from the Metatutor-KB. It tutors the different types of material for students of different proficiency levels. An advanced student is simply told that the answer is incorrect and given another chance, a student of average experience is given some hints before trying again, while the basic student is given a more extensive explanation before he selects another answer.

Summary

As a tutoring system SATS does not exhibit all of the behaviors described in Chapter Four. However, all eight knowledge bases contained in the system design have been exercised separately as well as in the integrated system. The Question Generator and Solver-KB, receiving the most emphasis, generates questions, answers, and distractors for facts, concepts, and principles. The following chapter describes the contributions from this project, makes recommendations for future research, and develops conclusions.

VI. Summary and Conclusions

Introduction

This study proposes a generic design for an ICAI system. With primary emphasis on the Question Generator and Solver-KB and secondary emphasis on the Student Model-KB, this design is implemented using the M.1 expert system building tool. This chapter discusses some contributions from this study, suggests areas for further research, and offers some conclusions.

Contributions

System Design and Testing. A major contribution from this study is the generic design of an ICAI system which presents lesson material, tests the student, diagnoses his errors, determines how to correct them, and then corrects them. By partitioning the knowledge needed by an ICAI system into eight knowledge bases, this design allows research to be focused on specific knowledge bases. Testing this design with the M.1 expert system tool demonstrates the feasibility of the design as well as the practicality of using an expert system building tool as a basis for an ICAI system.

Knowledge Representation Strategies. The development of different knowledge representation strategies allows SATS to test the student and then correct his errors. Since these strategies are tied only to the form of the knowledge (for example, terms or equations), they can be used for any

subject domain. Instructions for adding new terms or equations to SATS are given in Appendix A.

Question Generator and Solver-KB. The Question Generator and Solver-KB allows SATS to reason over a knowledge representation strategy (for terms or equations) to develop questions, answers, and distractors. As long as any equation is entered into SATS in the canonical form, the Question Generator and Solver-KB can perform the same function as it now does for the radar range equation.

Use of M.1. This is the first AFIT thesis to be based entirely on M.1. During this effort, several new techniques were developed which may be used for other projects. These techniques are noted in Appendix A. Experience with M.1 gained from this project provides a basis for evaluating M.1. Some advantages and limitations of M.1 are discussed later in the chapter.

Student Model-KB. The Student Model-KB allows SATS to better orient the tutoring to the student's needs. The initial student model is formed from questions asked of the student and then, based on the student's performance, the student model is updated throughout the session.

Areas for Future Research

SATS provides the framework for developing a tutoring system. Using its current knowledge representation strategies, SATS can test and tutor a student on terminology and on the relationships between the variables in a single

equation. Although SATS may be expanded using these current
knowledge representation strategies, additional knowledge
representation strategies are needed to make SATS function
as a true ICAI system. Two additional areas for research are
using external data bases to store the subject material and
converting SATS to a task-oriented CAI.

Knowledge Representation Strategies. The more
knowledge representation strategies available to SATS, the
greater its capability. The next level of complexity in
knowledge representation strategies is the semantic network.
Using a semantic network requires adapting the rule-based
format of M.1 to support a semantic network or converting
SATS to an environment more suited to using a semantic
network. An example of a semantic network of radar
characteristics is shown in Figure 6.1. Suggestions for
further research based on a semantic network are discussed
below in order of increasing difficulty.

Identifying and Describing Radar Characteristics.
Using this network, SATS can collect the characteristics of
a particular radar. It can then ask the student to
identify a particular radar from a list of characteristics,
or SATS can give the student a radar and ask him to describe
its characteristics. This idea can be further extended by
collecting the characteristics of more than one radar,
finding their common characteristics, and then asking the
student to identify these common characteristics.
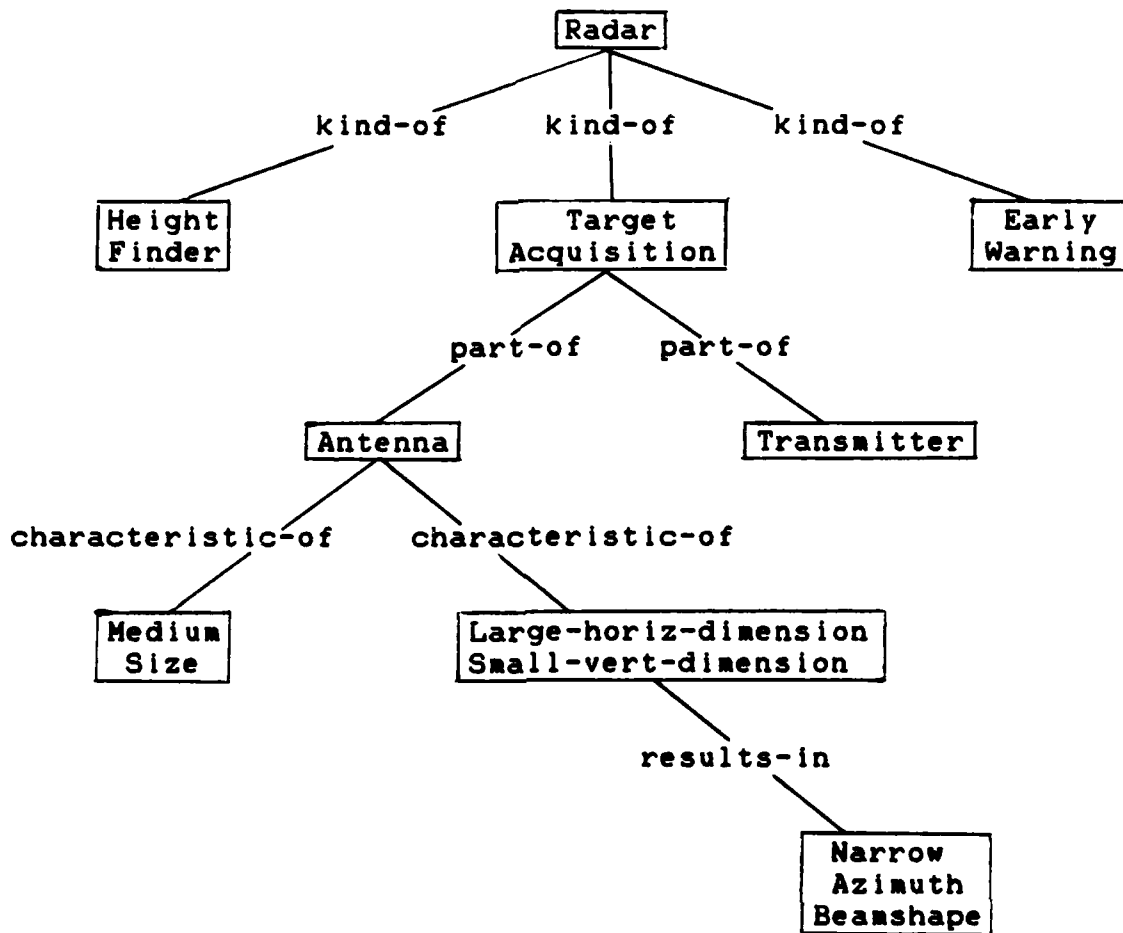
Figure 6.1 Semantic Network for Radars

Relational Questions.   The Question Generator and Solver-KB may use the semantic network to select two nodes in the semantic network.   With the three pieces of information represented by the two nodes and the relationship between them, SATS may give the student two of them and ask him to identify the third. Using Figure 6.1 as an example, SATS may select the two nodes as target acquisition radar and narrow azimuth beamshape and ask the student to identify the type of beamshape for a target

acquisition radar. The student is expected to either mentally trace the path between the nodes or be able to establish a direct link between the target acquisition radar and its beamshape. The Question Generator and Solver-KB may find distractors by finding the types of beamshapes for other radars.

Tutoring. After the student misses a question, the trace function of the inference process can lead the student from the question to the answer. Each time the student misses the question, the system shows him an intermediate node between the question and the answer. The first time the student misses the question, he is given information about the node next to the question; as he continues to miss the question, he is given information from nodes which are progressively closer to the answer until finally, if the student does not choose the correct answer before the tutor reaches the answer node, the system gives him the correct answer.

Extending the Current System. Although further development of most of the knowledge bases requires a new knowledge representation strategy such as described above, the equation handler may be extended as described below.

Converting Equations to Canonical Form. The current system requires equations to be entered into SATS in a canonical form before they can be used to develop questions. The system should allow equations to be entered

in another more general form and then have the system convert them to the required canonical format.

Sensitivity Analysis. The present equation handler performs sensitivity analysis between the dependent variable and a single independent variable. It could conduct sensitivity analysis on more than one independent variable by changing each independent variable, in turn, to see the effect on the dependent variable and then combining these effects into an overall effect.

Converting SATS to Task-Oriented Instruction. SATS now presents the student with information about radars. By changing the expertise component to accomodate an expert system, SATS can be converted to lead a student through the process of performing a task.

External Data Base. SATS, in its present form, contains the control function, the teaching modules, and the information specific to the subject area in a single computer file. As the lesson material grows, it needs to be separated from the control and teaching functions of SATS. Although M.1 permits external files to be used, the necessary interfaces need to be developed.

Conclusions

Background Required to Use M.1. A person need not have an extensive computer background to use M.1, but he should have a basic understanding of artificial intelligence techniques. Although the documentation for M.1 is fairly

extensive, the one-week course on M.1 offered by the Artificial Intelligence Laboratory is invaluable.

Limitations of M.1. Although M.1 offers a powerful developmental environment for SATS, its use in an actual training environment is doubtful. It is primarily limited by the size of its rule-base (no more than 1000 rules) and its speed. The computer code was changed in an effort to reduce the time required to generate questions, answers, and distractors for terms. This shortened the time by approximately one-half, but the time required was still approximately 35 seconds. Few students are willing to wait that long.

Continued Development of SATS. Training new personnel in an organization requires significant time and effort by the organization's experts. Intelligent Computer Aided Instruction systems provide a cost-effective alternative for training new personnel. Although the system design proposed in this thesis appears sound, the full development of SATS may take considerable effort by experienced knowledge engineers using large, hybrid system building tools.

Appendix A

Instructor's Guide for SATS

Introduction

This appendix explains the organization of SATS' computer code (listed in Appendix B), the procedures to modify what SATS tutors, and some of the useful programming techniques used.

Organization of SATS

Although all of the rules for SATS are in a single file, a particular rule may be found in its appropriate knowledge base. The knowledge bases are listed alphabetically according to the name of the knowledge base, and each rule within a particular knowledge base is listed according to the function it performs. To help the instructor locate a particular rule, a listing of the knowledge bases and the rules within each knowledge base is at the end of the computer code in Appendix B. Following is a brief description of each knowledge base.

The Control Rules Knowledge Base forms a step by step process for conducting the overall session. It is responsible for such things as establishing the initial student model and selecting the appropriate lesson.

The Diagnostics Knowledge Base determines the reason a student selects an incorrect answer.

The Lesson Knowledge Base contains the instructional text for the lesson.

The Metatutor Knowledge Base decides how to present the remedial tutoring according to the student's proficiency and the type of question he misses.

The Presentation Knowledge Base leads the student through a particular lesson. A lesson is divided into sections, and sections are further divided into slides. This knowledge base contains a list of the sections in a particular lesson and a list of the slides for each section.

The Question Generator and Solver Knowledge Base contains the terms and equations from which questions are developed. It also includes the procedures for developing the questions, answers, and distractors.

The Quizzing Knowledge Base is a series of questions asked of the student to obtain information to create the initial student model.

The Student Model Knowledge Base is a series of topics necessary to form the student model. Since the actual student model keeps changing, it is retained in working memory.

The Tutor Knowledge Base tutors the student depending on his proficiency level, the question he missed, and the reason he missed the question. It also governs the number of chances the student is given before he is told the correct answer.

The Miscellaneous Functions Knowledge Base contains rules which do not fit in any other knowledge base. These

include rules for finding the length of a list, removing items from a list, etc.

Changing the Information in SATS

Changing the Material Presented. As discussed above, the Presentation-KB operates through ordered lists. Modifying the sections within a lesson or the slides within a section merely requires changing the appropriate list.

Changing Terms. This information, which generates questions and answers, is in the Question Generator and Solver-KB. To change the terms on which the student is tested requires changing the list of terms in the 'symbols_for-section(terms)' rule. Additional changes must be made to the 'term_for_answer', 'answer_for_definition', and the 'answer_for_term' rules.

Changing Equations. This requires changes to two rules in the Question Generator and Solver-KB. The 'symbols_for-section(equations)' rule is changed to reflect the terms used in the equation. The equation is also entered in its canonical form into the 'define_equation' rule.

Programming Techniques

Speed Versus Flexibility. Speed and flexibility are two goals for SATS. For SATS to have flexibility requires using lists and recursive functions. However, both of these slow SATS speed. Therefore, the code for SATS balances these two goals. One technique uses both facts and recursive rules for some functions. An example is shown in

A - 3

the 'length' rule in the Miscellaneous Functions Knowledge Base. When a list is no longer than four elements, its length is determined by one of the 'length' facts. However, when the length exceeds four elements, a 'length' rule counts the number of elements in the list by groups of four until there are less than four remaining elements. These remaining elements are then counted by a 'length' fact. An instance where some flexibility is lost by not using recursion is in the 'make_question' rule located in the Question Generator and Solver-KB. Not using recursion results in limiting the maximimum number of questions which SATS can generate at any one time to three and in requiring three rules - rules to create one, two, or three questions - rather than a single recursive rule.

Message Function. Another programming technique is using a single 'message' rule for templates as well as for much of the text. Since different templates require passing different numbers of variables and text requires none, the first argument to the 'message' function is the type of message and the second argument is a list of the variables used.

Computer Code for SATS

```
/*                      RULES FOR SATS                      */

initialdata = [lesson_complete].


/*                 CONFIGURATION META-FACTS                 */

configuration(startup) = go.

configuration(banner) = [
'Welcome to the Signal Analyst Tutoring System.  I will instruct
you in some radar principles and then ask you a few questions
about these principles.  If you miss a question, I will try to',nl,
'tell you why you are wrong and then let you try again.  To answer
a question, type in the number preceding the answer.  For example,
if the answer is ',nl, nl,
'        ''2.  range'', you would type in ''2'' ' ,nl,nl,
'If you want to quit before the end of the lesson, type
<CTRL-BREAK>.  That will return you to the M.1 level.  If you
type <CTRL-BREAK> again, it will return you to the DOS level.',
' If you have any questions, you may need to see your instructor.',
nl].
```

```
/* ============================================================ */
/* ============================================================ */
/*                 CONTROL RULES KNOWLEDGE BASE                 */
/* ============================================================ */
/* ============================================================ */


/*                         LESSON                              */

automaticmenu(lesson).
enumeratedanswers(lesson).

question(lesson) =
[nl, 'Which lesson shall we cover?'].

legalvals(lesson) = [radar_range_equation,
                     fundamental_relationships,
                     decibels].


/*                     LESSON_COMPLETE                         */


if   seed = NUMBER and
     initial_student_model is sought and
     lesson = LESSON and
     sections(LESSON) = LIST_OF_SECTIONS and
     each-section_type-of-LIST_OF_SECTIONS-of-LESSON is complete
          and
     finished_with_lesson
then lesson_complete.
```

```
/* =============================================================== */
/* =============================================================== */
/*                   DIAGNOSIS KNOWLEDGE BASE                      */
/* =============================================================== */
/* =============================================================== */


/*                        DIAGNOSIS                               */

nocache(diagnosis(V,W,X,Y,Z)).


/* DEFINITIONS */

diagnosis(TERM,CORRECT_ANSWER,STUDENT_ANSWER,definitions,N) =
          not_know_term.


/* QUALITATIVE RELATIONSHIPS */

diagnosis(TERM,CORRECT_ANSWER,STUDENT_ANSWER,
          qualitative_relationships,N) =
          not_understand_qualitative_relationship.


/* QUANTITATIVE RELATIONSHIPS */

if   CORRECT_ANSWER == [C_ANSWER,C_RELATED,C_EXPONENT] and
     STUDENT_ANSWER == [S_ANSWER,S_RELATED,S_EXPONENT] and
     C_EXPONENT == S_EXPONENT and
     not(C_RELATED == S_RELATED)
then diagnosis(TERM,CORRECT_ANSWER,STUDENT_ANSWER,
          quantitative_relationships,N) = relationship.

if   CORRECT_ANSWER == [C_ANSWER,C_RELATED,C_EXPONENT] and
     STUDENT_ANSWER == [S_ANSWER,S_RELATED,S_EXPONENT] and
     C_RELATED == S_RELATED and
     not(C_EXPONENT == S_EXPONENT)
then diagnosis(TERM,CORRECT_ANSWER,STUDENT_ANSWER,
          quantitative_relationships,N) = factor.

if   CORRECT_ANSWER == [C_ANSWER,C_RELATED,C_EXPONENT] and
     STUDENT_ANSWER == [S_ANSWER,S_RELATED,S_EXPONENT] and
     not(C_ANSWER == S_ANSWER) and
     not(C_EXPONENT == S_EXPONENT)
then diagnosis(TERM,CORRECT_ANSWER,STUDENT_ANSWER,
          quantitative_relationships,N) = both_wrong.


/* TERMS */

diagnosis(TERM,CORRECT_ANSWER,STUDENT_ANSWER,terms,N) =
          not_know_term.
```

```prolog
/* ================================================================ */
/* ================================================================ */
/*                    LESSON KNOWLEDGE BASE                         */
/* ================================================================ */
/* ================================================================ */


/*                        PRESENTATION                             */

nocache(presentation(X,Y)).

presentation(introduce_radar_range_equation,why_important) = [nl,
'We will be looking at the radar range equation which relates
characteristics of transmitter, receiver, antenna, target, and
environment.',' I will tell you why it is important, explain
what the symbols mean, and how the terms are related. We will
do this for several forms of the radar range equation.',nl].

presentation(equation_1,motivation) = [nl,
'The first radar equation is' ,nl,

'            ((P(t)) * (G**2) * (lambda**2) * sigma)
      P(r) = -------------------------------------------
                    (((4*pi)**3) * (R**4))'            ,nl,

'This equation is important because it shows how the power
received by the radar is affected by several factors.',' In the
above equation, a single asterisk means to multiply the terms
while a double asterisk means to raise the term to a power.  The
next slide will define the factors in the equation.',nl].

presentation(equation_1,terms) = [nl,
'The following terms make up the basic radar range equation.',
nl,nl,
'  P(r) is the power received by the radar.' ,nl,
'  P(t) is the power transmitted by the radar.' ,nl,
'  G is the gain of the antenna.' ,nl,
'  Lambda is the wavelength.' ,nl,
'  Sigma is the cross sectional area of the target.' ,nl,
'  R is the range of the target from the radar.' ,nl].

presentation(equation_1,definitions) = [nl,'Equation 1
          definitions.',nl].

presentation(equation_1,qualitative_relationships) = [nl,
'You will note that the power received by the radar increases
when a term in the numerator increases and it decreases as a
term in the denominator increases.',nl].
```

presentation(equation_1,quantitative_relationships) = [nl,
'The power received at the radar varies as the parameters on
the right hand side of the equation change. The effect of a
parameter on the power received depends on the amount the term
is changed and the exponential factor of the term.',' For
instance, if the gain is doubled, the power received is
quadrupled.',nl].

presentation(equation_2,motivation) = [nl,
'The next form of the radar range equation is',nl,
'

$$R = \left[\frac{P(t) * (G**2) * (lambda**2) * sigma}{(((4*pi)**3) * P(r))} \right]^{(1/4)}$$

' ,nl,

'This equation shows how the detection range is affected by
several different parameters. Note that range is related to
these terms by the fourth root. There are no new terms for
this equation.',nl].

presentation(equation_2,qualitative_relationships) = [nl,
'Again, note that the range increases when a paramter in the
numerator increases and decreases when a parameter in the
denominator increases.',nl].

presentation(equation_2,quantitative_relationships) = [nl,
'Remember that range is affected by the fourth root of the
other parameters.',nl].

presentation(equation_3,motivation) = [nl,
'The last form of the radar range equation which we will
examine is',nl,
'

$$R = \left[\frac{P(t) * (A(e)**2) * sigma) * (t(ot))}{((4*pi) * S * (lambda**2))} \right]^{(1/4)}$$

' ,nl,

'This form of the equation considers the number of radar pulses
striking the target. As the number of pulses received from the
target increases, the greater the detection range.',nl].

presentation(equation_3,terms) = [nl,
'Two new terms are in this equation.',nl,nl,
' A(e) is the effective antenna area.',nl,
' t(ot) is the length of time the radar beam is on the target.',
nl].

presentation(equation_3,definitions) = [nl, 'Equation 3
definitions.',nl].

presentation(equation_3,qualitative_relationships) = [nl,
'Again, terms in the numerator cause range to increase while
terms in the denominator cause it to decrease.',nl].

```
presentation(equation_3,quantitative_relationships) = [nl,
'The quantitative relationships in this equation are similar to
those in the previous equation.',nl].

presentation(ANY_SECTION,dummy_slides) = [nl,
'This lesson is not installed.  Please type ''abort.'' to return
to M.1 and then type in ''go.'' to start again.'].
```

```
/* ================================================================= */
/* ================================================================= */
/*                    METATUTOR KNOWLEDGE BASE                       */
/* ================================================================= */
/* ================================================================= */


/*                         METATUTOR                                 */

nocache(metatutor(X,Y,Z)).
nocache(metatutor(W,X,Y,Z)).
nocache(metatutor(V,W,X,Y,Z)).


/* QUALTITATIVE RELATIONSHIPS */

metatutor(TERM,not_understand_qualitative_relationship,direct,N) =
        TERM-in_numerator-for_qualitative_relationship.

metatutor(TERM,not_understand_qualitative_relationship,inverse,N) =
        TERM-in_denominator-for_qualitative_relationship.

metatutor(TERM,not_understand_qualitative_relationship,unrelated,N) =
        TERM-unrelated-for_qualitative_relationship.


/* QUANTITATIVE RELATIONSHIPS */

metatutor(TERM,both_wrong,LEFT_HAND_SIDE,[ANSWER,RELATED,FACTOR],N) =
        TERM-is-RELATED-to-LEFT_HAND_SIDE-by-FACTOR-
        for_quantitative_relationship.

metatutor(TERM,factor,LEFT_HAND_SIDE,[ANSWER,RELATED,FACTOR],N) =
        FACTOR-for-TERM-for_quantitative_relationship_with-
        LEFT_HAND_SIDE.

metatutor(TERM,relationship,LEFT_HAND_SIDE,[ANSWER,direct,FACTOR],N)
        TERM-in_numerator-for_quantitative_relationship_with-
        LEFT_HAND_SIDE.

metatutor(TERM,relationship,LEFT_HAND_SIDE,[ANSWER,inverse,FACTOR],N)
        TERM-in_denominator-for_quantitative_relationship_with-
        LEFT_HAND_SIDE.

metatutor(TERM,relationship,LEFT_HAND_SIDE,[ANSWER,unrelated,FACTOR],
        N) = TERM-unrelated-for_quantitative_relationship_with-
        LEFT_HAND_SIDE.


/* TERMS */

metatutor(not_know_term,STUDENT_ANSWER,N) = define-STUDENT_ANSWER.
```

```
/* ================================================================ */
/* ================================================================ */
/*                  PRESENTATION KNOWLEDGE BASE                     */
/* ================================================================ */
/* ================================================================ */


/*                        SECTIONS                                 */

nocache(sections(X)).

sections(radar_range_equation) = [introduce_radar_range_equation,
                                  equation_1,
                                  equation_2,
                                  equation_3].

sections(fundamental_relationships) =
        [introduce_fundamental_relationships,
        wavelength_frequency,
        prf_pri].

sections(decibels) = [introduce_decibels,
                      decibels_power,
                      decibels_voltage].


/*                        SLIDES                                   */

nocache(slides(LESSON,SECTION)).

slides(radar_range_equation,introduce_radar_range_equation) =
        [why_important].

slides(radar_range_equation,equation_1) =
        [motivation,terms,qualitative_relationships,
        quantitative_relationships].

slides(radar_range_equation,equation_2) =
        [motivation,qualitative_relationships,
        quantitative_relationships].

slides(radar_range_equation,equation_3) =
        [motivation,terms,qualitative_relationships,
        quantitative_relationships].

slides(X,Y) = [dummy_slides].
```

```
/* ================================================================ */
/* ================================================================ */
/*        QUESTION GENERATOR AND SOLVER KNOWLEDGE BASE              */
/* ================================================================ */
/* ================================================================ */


/*                     ANSWER_FOR_DEFINITION                        */

nocache(answer_for_definition(X)).

answer_for_definition('R') = [nl,
          'The radial distance from a radar to a target or other
          object.'].
answer_for_definition('P(t)') = [nl,
          'The power transmitted by the radar.'].
answer_for_definition('P(r)') = [nl,
          'The power received by the radar.'].
answer_for_definition('G') = [nl,
          'The ratio of the power of the radiation in a given
          direction to the power of the radiation that would
          be produced in that direction if the same input power
          were applied to a hypothetical isotropic antenna.'].
answer_for_definition('A(e)') = [nl,
          'The physical area times aperture efficiency.'].
answer_for_definition('t(ot)') = [nl,
          'The time during which a target is continuouly in the
          mainlobe of a searching radar''s antenna on any one
          scan of the antenna.'].
answer_for_definition(sigma) = [nl,
          'A factor relating the power of the radio waves that
          a radar target scatters back in the direction of the
          radar to the power density of the radar''s transmitted
          waves at the target''s range.'].
answer_for_definition(lambda) = [nl,
          'The distance between successive ''crests'' or between
          points at which the intensity of the field goes through
          zero in the same direction.'].
```

```
/*              ANSWER_FOR_QUALITATIVE_RELATIONSHIPS              */

nocache(answer_for_qualitative_relationships(SECTION,TERM)).

if    define_equation(SECTION) = CANONICAL_EQUATION and
      related(CANONICAL_EQUATION,TERM) = RELATIONSHIP
then  answer_for_qualitative_relationships(SECTION,TERM) =
          RELATIONSHIP..


/*              ANSWER_FOR_QUANTITATIVE_RELATIONSHIPS             */

/*    Restrictions: Amount a parameter is changed >= 1.
                  : A power must be an integer.
                  : A root multiplied by 2 must eventually = 1.
                  : Quotient can be used only if it is the first
                    operator or, if root is the first operator,
                    quotient must be the next operator.         */


/*              ANSWER_FOR_QUANTITATIVE_RELATIONSHIP             */

nocache(answer_for_quantitative_relationship(X,Y,Z)).

if    quantitative(EQUATION,TERM) = [RELATED,ROOT_EXPONENT,
          ROOT_EXPONENT_VALUE] and
      convert_root_to_exponent(ROOT_EXPONENT,ROOT_EXPONENT_VALUE) =
          EXPONENT_VALUE and
      translate(CHANGE_TERM_VALUE,RELATED,EXPONENT_VALUE) =
          CORRECT_ANSWER and
      answers_quantitative_relationships(CHANGE_TERM_VALUE,
          EXPONENT_VALUE,CORRECT_ANSWER) = ORDERED_ANSWERS
then  answer_for_quantitative_relationship(EQUATION,TERM,
          CHANGE_TERM_VALUE) = [CORRECT_ANSWER,ORDERED_ANSWERS].


/*              ANSWERS_QUANTITATIVE_RELATIONSHIPS               */

nocache(answers_quantitative_relationships(X,Y,Z)).

if    different_from-EXPONENT_VALUE = DISTRACTOR_NUMBER and
      translate(CHANGE_TERM_VALUE,direct,DISTRACTOR_NUMBER) =
          WRONG_ANSWER and
      order_answers(CORRECT_ANSWER,WRONG_ANSWER) = ORDERED_ANSWERS
then  answers_quantitative_relationships(CHANGE_TERM_VALUE,
          EXPONENT_VALUE,CORRECT_ANSWER) = ORDERED_ANSWERS.
```

```
/*                        ANSWER_FOR_TERM                        */

nocache(answer_for_term(X)).

answer_for_term('R') = range.
answer_for_term('P(t)') = power_transmitted.
answer_for_term('P(r)') = power_received.
answer_for_term('G') = gain.
answer_for_term('A(e)') = effective_antenna_area.
answer_for_term('t(ot)') = time_on_target.
answer_for_term('Sigma') = radar_cross_section.
answer_for_term('Lambda') = wavelength.


/*                    ANSWERS_DEFINITIONS                        */

nocache(answers_definitions(X)).

if    answer_for_definition(A) = E1 and
      answer_for_definition(B) = E2 and
      answer_for_definition(C) = E3 and
      answer_for_definition(D) = E4
then answers_definitions([A,B,C,D]) = [E1,E2,E3,E4].


/*            ANSWERS_QUALITATIVE_RELATIONSHIPS                  */

answers_qualitative_relationships(qualitative_relationships) =
                                   [direct,inverse,unrelated].


/*                      ANSWERS_TERMS                           */

nocache(answers_terms(X)).

if    answer_for_term(A) = E1 and
      answer_for_term(B) = E2 and
      answer_for_term(C) = E3 and
      answer_for_term(D) = E4
then answers_terms([A,B,C,D]) = [E1,E2,E3,E4].
```

```
enumeratedanswers(ask_a_question(SLIDE,ARGUMENT_LIST)).


/* DEFINITIONS */

question(ask_a_question(definitions,[TERM]))=  [nl,
'How would you define ',TERM, '?'].

automaticmenu(ask_a_question(definitions,TERM)).


/* QUALITATIVE RELATIONSHIP */

question(ask_a_question(qualitative_relationships,
          [TERM,LEFT_HAND_SIDE])) = [nl,
'What is the relationship between ',LEFT_HAND_SIDE, ' and ',
          TERM, '.'].

legalvals(ask_a_question(qualitative_relationships,
          ARGUMENT_LIST)) = [direct,inverse,unrelated].
automaticmenu(ask_a_question(qualitative_relationships,
          ARGUMENT_LIST)).


/* QUANTITATIVE RELATIONSHIPS */

question(ask_a_question(quantitative_relationships,
          [TERM,LEFT_HAND_SIDE,CHANGE_TERM_VALUE,
          [[ANSWER_1:REST_1],[ANSWER_2:REST_2],
          [ANSWER_3:REST_3],[ANSWER_4:REST_4]]])) = [nl,
'How does increasing ',TERM, ' by a factor of ',CHANGE_TERM_VALUE,'
affect ',LEFT_HAND_SIDE,nl,nl,
'          1.  ',ANSWER_1,nl,
'          2.  ',ANSWER_2,nl,
'          3.  ',ANSWER_3,nl,
'          4.  ',ANSWER_4,nl].


/* TERMS */

question(ask_a_question(terms,[TERM]))=  [nl,
'What does the symbol ',TERM, ' represent?'].

automaticmenu(ask_a_question(terms,TERM)).
```

```
nocache(check_answer(X,Y,Z) is satisfactory).


/* QUALTITATIVE RELATIONSHIPS */

if   NEW_STUDENT_ANSWER == CORRECT_ANSWER and
     message(confirm_answer,CORRECT_ANSWER)
then check_answer(CORRECT_ANSWER,NEW_STUDENT_ANSWER,
          qualitative_relationships) is satisfactory.


/* QUANTITATIVE RELATIONSHIPS */

if   NEW_STUDENT_ANSWER == CORRECT_ANSWER and
     CORRECT_ANSWER == [ANSWER,RELATED,EXPONENT_VALUE] and
     message(confirm_answer,ANSWER)
then check_answer(CORRECT_ANSWER,NEW_STUDENT_ANSWER,
          quantitative_relationships) is satisfactory.


/* TERMS */

if   NEW_STUDENT_ANSWER == CORRECT_ANSWER and
     message(confirm_answer,CORRECT_ANSWER)
then check_answer(CORRECT_ANSWER,NEW_STUDENT_ANSWER,terms)
          is satisfactory.


/*                          DEFINE_EQUATION                       */

define_equation(equation_1) =
  [quotient,[product,[product,'P(t)',[exponent,'G',2]],
                      [product,[exponent,'Lambda',2],'Sigma']],
          [product,[exponent,[product,4,pi],3],
                      [exponent,'R',4]]].

define_equation(equation_2) =
  [root,[quotient,[product,[product,'P(t)',[exponent,'G',2]],
                      [product,[exponent,'Lambda',2],'Sigma']],
               [product,[exponent,[product,4,pi],3],
                      'P(r)']],
        4].

define_equation(equation_3) =
  [root,[quotient,[product,[product,'P(t)',[exponent,'A(e)',2]],
                      [product,'Sigma','t(ot)']],
               [product,[product,4,pi],
                      [product,'S',[exponent,'Lambda',2]]]],
        4].
```

```
/*              DIFFERENT_FROM-EXPONENT_VALUE              */

nocache(different_from-X).

if    list_of_numbers-2 = CHOICE_OF_NUMBERS and
      remove(EXPONENT_VALUE,CHOICE_OF_NUMBERS) =
          NEW_CHOICE_OF_NUMBERS and
      pick(NEW_CHOICE_OF_NUMBERS) = DISTRACTOR_NUMBER
then different_from-EXPONENT_VALUE = DISTRACTOR_NUMBER.

if    list_of_numbers-2 = CHOICE_OF_NUMBERS and
      pick(NEW_CHOICE_OF_NUMBERS) = DISTRACTOR_NUMBER
then different_from-EXPONENT_VALUE = DISTRACTOR_NUMBER.


/*              LEFT_HAND_SIDE_FOR-EQUATION_X              */

nocache(left_hand_side_for-X).

left_hand_side_for-equation_1 = 'P(r)'.

left_hand_side_for-equation_2 = 'R'.

left_hand_side_for-equation_3 = 'R'.


/*              LIST_OF_NUMBERS-X              */

nocache(list_of_numbers-X).

list_of_numbers-1 = [2.0,3.0,4.0].
list_of_numbers-2 = [0.25,0.5,1,2,4].


/*              MAKE_QUESTIONS              */

nocache(make_questions(SECTION,TERMS,SLIDE,NUMBER)).

if    single_question(SECTION,TERMS,SLIDE) = [T1|R1]
then make_questions(SECTION,TERMS,SLIDE,1) = [T1|R1].

if    single_question(SECTION,TERMS,SLIDE) = [T1|R1] and
      remove(T1,TERMS) = TERMS2 and
      single_question(SECTION,TERMS2,SLIDE) = [T2|R2]
then make_questions(SECTION,TERMS,SLIDE,2) = [[T1|R1],[T2|R2]].

if    single_question(SECTION,TERMS,SLIDE) = [T1|R1] and
      remove(T1,TERMS) = TERMS2 and
      single_question(SECTION,TERMS2,SLIDE) = [T2|R2] and
      remove(T2,TERMS2) = TERMS3 and
      single_question(SECTION,TERMS3,SLIDE) = [T3|R3]
then make_questions(SECTION,TERMS,SLIDE,3) =
          [[T1|R1],[T2|R2],[T3|R3]].
```

/* DEFINITIONS */

```
if    do(reset ask_a_question(definitions,TERM)) and
      ask_a_question(definitions,TERM) = NEW_STUDENT_ANSWER
then new_answer(definitions,N,TERM) = NEW_STUDENT_ANSWER.
```


/* QUALITATIVE RELATIONSHIPS */

```
if    do(reset ask_a_question(qualitative_relationships,
           [TERM,LEFT_HAND_SIDE])) and
      ask_a_question(qualitative_relationships,[TERM,
           LEFT_HAND_SIDE]) = NEW_STUDENT_ANSWER
then new_answer(qualitative_relationships,N,[TERM,
           LEFT_HAND_SIDE]) = NEW_STUDENT_ANSWER.
```


/* QUANTITATIVE RELATIONSHIPS */

```
if    do(reset ask_a_question(quantitative_relationships,
           [TERM,LEFT_HAND_SIDE,CHANGE_TERM_VALUE,ANSWERS])) and
      ask_a_question(quantitative_relationships,[TERM,LEFT_HAND_SIDE,
           CHANGE_TERM_VALUE,ANSWERS]) = NEW_STUDENT_ANSWER
then new_answer(quantitative_relationships,N,[TERM,LEFT_HAND_SIDE,
           CHANGE_TERM_VALUE,ANSWERS]) = NEW_STUDENT_ANSWER.
```


/* TERMS */

```
if    do(reset ask_a_question(terms,TERM)) and
      ask_a_question(terms,TERM) = NEW_STUDENT_ANSWER
then new_answer(terms,N,TERM) = NEW_STUDENT_ANSWER.
```

```
/*                       NUMBER_OF_QUESTIONS                      */

nocache(number_of_questions(SLIDE,PROFICIENCY)).

number_of_questions(terms,basic) = 3.
number_of_questions(terms,average) = 2.
number_of_questions(terms,advanced) = 1.

number_of_questions(definitions,basic) = 3.
number_of_questions(definitions,average) = 2.
number_of_questions(definitions,advanced) = 1.

number_of_questions(qualitative_relationships,basic) = 2.
number_of_questions(qualitative_relationships,average) = 1.
number_of_questions(qualitative_relationships,advanced) = 1.

number_of_questions(quantitative_relationships,basic) = 2.
number_of_questions(quantitative_relationships,average) = 1.
number_of_questions(quantitative_relationships,advanced) = 1.


/*                          ORDER_ANSWERS                         */

nocache(order_answers(X,Y)).

if    CORRECT_ANSWER == [RIGHT_CHANGE,RELATED,EXPONENT_VALUE] and
      convert_relationship(RELATED) = INVERTED_RELATIONSHIP and
      round_number((1 / RIGHT_CHANGE),3) = INVERSE_RIGHT_CHANGE and
      DISTRACTOR_2 == [INVERSE_RIGHT_CHANGE,INVERTED_RELATIONSHIP,
          EXPONENT_VALUE] and
      WRONG_ANSWER == [WRONG_CHANGE,direct,WRONG_EXPONENT_VALUE] and
      round_number((1 / WRONG_CHANGE),3) = INVERSE_WRONG_CHANGE and
      DISTRACTOR_4 == [INVERSE_WRONG_CHANGE,inverse,
          WRONG_EXPONENT_VALUE] and
      put_direct_answer_first(CORRECT_ANSWER,DISTRACTOR_2) =
                              [ANSWER_1,ANSWER_2] and
      random(4) = RANDOM_NUMBER and
      scramble-RANDOM_NUMBER-for-[ANSWER_1,ANSWER_2,WRONG_ANSWER,
          DISTRACTOR_4] = ANSWERS
then order_answers(CORRECT_ANSWER,WRONG_ANSWER) = ANSWERS.


/*                         PUT_DIRECT_FIRST                       */

nocache(put_direct_answer_first(X,Y)).

if    CORRECT_ANSWER == [RIGHT_CHANGE,direct,EXPONENT_VALUE]
then put_direct_answer_first(CORRECT_ANSWER,DISTRACTOR_2) =
          [CORRECT_ANSWER,DISTRACTOR_2].

if    CORRECT_ANSWER == [RIGHT_CHANGE,inverse,EXPONENT_VALUE]
then put_direct_answer_first(CORRECT_ANSWER,DISTRACTOR_2) =
          [DISTRACTOR_2,CORRECT_ANSWER].
```

```
nocache(quantitative(X,Y)).

if    define_equation(EQUATION) = [quotient,NUMERATOR,DENOMINATOR]
          and
      contains(NUMERATOR,TERM) = [ROOT_EXPONENT,ROOT_EXPONENT_VALUE]
then  quantitative(EQUATION,TERM) = [direct,ROOT_EXPONENT,
          ROOT_EXPONENT_VALUE].


if    define_equation(EQUATION) = [quotient,NUMERATOR,DENOMINATOR]
          and
      contains(DENOMINATOR,TERM) = [ROOT_EXPONENT,ROOT_EXPONENT_VALUE]
then  quantitative(EQUATION,TERM) = [inverse,ROOT_EXPONENT,
          ROOT_EXPONENT_VALUE].


if    define_equation(EQUATION) = [root,[quotient,NUMERATOR,
          DENOMINATOR],ROOT_VALUE] and
      contains(NUMERATOR,TERM) = [ROOT_EXPONENT,
          ROOT_EXPONENT_VALUE] and
      combine(root,ROOT_VALUE,ROOT_EXPONENT,ROOT_EXPONENT_VALUE) =
          [POWER_ROOT,POWER_ROOT_VALUE]
then  quantitative(EQUATION,TERM) = [direct,POWER_ROOT,
          POWER_ROOT_VALUE].


if    define_equation(EQUATION) = [root,[quotient,NUMERATOR,
          DENOMINATOR],ROOT_VALUE] and
      contains(DENOMINATOR,TERM) = [ROOT_EXPONENT,
          ROOT_EXPONENT_VALUE] and
      combine(root,ROOT_VALUE,ROOT_EXPONENT,ROOT_EXPONENT_VALUE) =
          [POWER_ROOT,POWER_ROOT_VALUE]
then  quantitative(EQUATION,TERM) = [inverse,ROOT_EXPONENT,
          POWER_ROOT_VALUE].


if    define_equation(EQUATION) = [FIRST_OPERATOR,FIRST_ARGUMENT,
          SECOND_ARGUMENT] and
      contains(FIRST_ARGUMENT,TERM) = [SECOND_OPERATOR,SECOND_VALUE]
          and
      combine(FIRST_ARGUMENT,SECOND_ARGUMENT,SECOND_OPERATOR,
          SECOND_VALUE) = [NEW_OPERATOR,NEW_VALUE]
then  quantitative(EQUATION,TERM) = [direct,NEW_OPERATOR,NEW_VALUE].


if    define_equation(EQUATION) = [FIRST_OPERATOR,FIRST_ARGUMENT,
          SECOND_ARGUMENT] and
      contains(SECOND_ARGUMENT,TERM) = [SECOND_OPERATOR,
          SECOND_VALUE] and
      combine(FIRST_OPERATOR,SECOND_ARGUMENT,SECOND_OPERATOR,
          SECOND_VALUE) = [NEW_OPERATOR,NEW_VALUE]
then  quantitative(EQUATION,TERM) = [direct,NEW_OPERATOR,NEW_VALUE].


if    define_equation(EQUATION) = EXPRESSION and
      contains(EXPRESSION,TERM) = [OPERATOR,VALUE]
then  quantitative(EQUATION,TERM) = [direct,OPERATOR,VALUE].
```

```
/*                        QUIZ_STUDENT                        */

nocache(quiz_student(SECTION,TERMS,SLIDE)).

if    student_model-SLIDE = PROFICIENCY and
      number_of_questions(SLIDE,PROFICIENCY) = NUMBER_OF_QUESTIONS
            and
      make_questions(SECTION,TERMS,SLIDE,NUMBER_OF_QUESTIONS) =
            LIST_OF_QUESTIONS and
      each-ask_question-of-LIST_OF_QUESTIONS-of-SLIDE is complete
            and
      student_model_updated(SLIDE) is sought and
      do(reset ask_a_question(SLIDE,QUESTION_ARGUMENTS)) and
      do(reset new_answer(SLIDE,N,QUESTION_ARGUMENTS)) and
      do(reset number_of_correct_answers(SLIDE))
then quiz_student(SECTION,TERMS,SLIDE).


/*                         SCRAMBLE                          */

nocache(scramble-X-for-Y).

scramble-1-for-[ANS_1,ANS_2,ANS_3,ANS_4] =
          [ANS_1,ANS_3,ANS_2,ANS_4].

scramble-2-for-[ANS_1,ANS_2,ANS_3,ANS_4] =
          [ANS_3,ANS_1,ANS_4,ANS_2].

scramble-3-for-[ANS_1,ANS_2,ANS_3,ANS_4] =
          [ANS_4,ANS_2,ANS_1,ANS_3].

scramble-4-for-[ANS_1,ANS_2,ANS_3,ANS_4] =
          [ANS_2,ANS_4,ANS_3,ANS_1].
```

```
nocache(single-ask_question-of-ARGUMENT_LIST-of-SLIDE is complete).
noautomaticquestion(single-ask_question-of-ARGUMENT_LIST-of-SLIDE
        is complete).
```

/* DEFINITIONS */

```
if    do(addz temp_list:legalvals(ask_a_question(definitions,
            [TERM])) = ORDERED_ANSWERS) and
      ask_a_question(definitions,[TERM]) = STUDENT_ANSWER and
      STUDENT_ANSWER == CORRECT_ANSWER and
      message(confirm_answer,CORRECT_ANSWER)
then  single-ask_question-of-[TERM,CORRECT_ANSWER,
            ORDERED_ANSWERS]-of-definitions is complete.

if    ask_a_question(definitions,[TERM]) = STUDENT_ANSWER and
      not(STUDENT_ANSWER == CORRECT_ANSWER) and
      tutoring(TERM,CORRECT_ANSWER,STUDENT_ANSWER,terms) is
            satisfactory
then  single-ask_question-of-[TERM,CORRECT_ANSWER,
            ORDERED_ANSWERS]-of-definitions is complete.

if    message(tell_answer,CORRECT_ANSWER)
then  single-ask_question-of-[TERM,CORRECT_ANSWER,
            ORDERED_ANSWERS]-of-definitions is complete.
```

/* QUALITATIVE RELATIONSHIPS */

```
if    ask_a_question(qualitative_relationships,
            [TERM,LEFT_HAND_SIDE]) = STUDENT_ANSWER and
      STUDENT_ANSWER == CORRECT_ANSWER and
      message(confirm_answer,CORRECT_ANSWER)
then  single-ask_question-of-[TERM,LEFT_HAND_SIDE,CORRECT_ANSWER,
            ORDERED_ANSWERS]-of-qualitative_relationships is complete.

if    ask_a_question(qualitative_relationships,
            [TERM,LEFT_HAND_SIDE]) = STUDENT_ANSWER and
      not(STUDENT_ANSWER == CORRECT_ANSWER) and
      tutoring(TERM,LEFT_HAND_SIDE,CORRECT_ANSWER,STUDENT_ANSWER,
            qualitative_relationships) is satisfactory
then  single-ask_question-of-[TERM,LEFT_HAND_SIDE,CORRECT_ANSWER,
            ORDERED_ANSWERS]-of-qualitative_relationships is complete.

if    message(tell_answer,CORRECT_ANSWER)
then  single-ask_question-of-[TERM,LEFT_HAND_SIDE,CORRECT_ANSWER,
            ORDERED_ANSWERS]-of-qualitative_relationships is complete.
```

```
/* QUANTITATIVE RELATIONSHIPS */

if    do(addz temp_list:legalvals(ask_a_question
          (quantitative_relationships,[TERM,LEFT_HAND_SIDE,
          CHANGE_TERM_VALUE,ORDERED_ANSWERS])) = ORDERED_ANSWERS)
          and
      ask_a_question(quantitative_relationships,[TERM,LEFT_HAND_SIDE,
          CHANGE_TERM_VALUE,ORDERED_ANSWERS]) = STUDENT_ANSWER and
      STUDENT_ANSWER == CORRECT_ANSWER and
      CORRECT_ANSWER == [ANSWER,RELATED,EXPONENT_VALUE] and
      message(confirm_answer,ANSWER)
then  single-ask_question-of-[TERM,LEFT_HAND_SIDE,CHANGE_TERM_VALUE,
          CORRECT_ANSWER,ORDERED_ANSWERS]-of-
          quantitative_relationships is complete.

if    ask_a_question(quantitative_relationships,[TERM,LEFT_HAND_SIDE,
          CHANGE_TERM_VALUE,ORDERED_ANSWERS]) = STUDENT_ANSWER and
      not(STUDENT_ANSWER == CORRECT_ANSWER) and
      tutoring(TERM,LEFT_HAND_SIDE,CHANGE_TERM_VALUE,CORRECT_ANSWER,
          STUDENT_ANSWER,ORDERED_ANSWERS,quantitative_relationships)
          is satisfactory
then  single-ask_question-of-[TERM,LEFT_HAND_SIDE,CHANGE_TERM_VALUE,
          CORRECT_ANSWER,ORDERED_ANSWERS]-of-
          quantitative_relationships is complete.

if    CORRECT_ANSWER == [ANSWER,RELATED,EXPONENT_VALUE] and
      message(tell_answer,ANSWER)
then  single-ask_question-of-[TERM,LEFT_HAND_SIDE,CHANGE_TERM_VALUE,
          CORRECT_ANSWER,ORDERED_ANSWERS]-of-
          quantitative_relationships is complete.


/* TERMS */

if    do(addz temp_list:legalvals(ask_a_question(terms,[TERM])) =
          ORDERED_ANSWERS) and
      ask_a_question(terms,[TERM]) = STUDENT_ANSWER and
      STUDENT_ANSWER == CORRECT_ANSWER and
      message(confirm_answer,CORRECT_ANSWER)
then  single-ask_question-of-[TERM,CORRECT_ANSWER,
          ORDERED_ANSWERS]-of-terms is complete.

if    ask_a_question(terms,[TERM]) = STUDENT_ANSWER and
      not(STUDENT_ANSWER == CORRECT_ANSWER) and
      tutoring(TERM,CORRECT_ANSWER,STUDENT_ANSWER,terms)
          is satisfactory
then  single-ask_question-of-[TERM,CORRECT_ANSWER,
          ORDERED_ANSWERS]-of-terms is complete.

if    message(tell_answer,CORRECT_ANSWER)
then  single-ask_question-of-[TERM,CORRECT_ANSWER,
          ORDERED_ANSWERS]-of-terms is complete.
```

```
nocache(single_question(SECTION,TERMS,SLIDE)).


/* DEFINITIONS */

if    multipick(TERMS,4) = LIST_OF_TERMS and
      answers_definitions(LIST_OF_TERMS) = ANSWERS and
      pick(LIST_OF_TERMS) = TERM and
      answer_for_definition(TERM) = CORRECT_ANSWER and
      display([nl,nl,'I''m still working.',nl,nl])
then  single_question(SECTION,TERMS,definitions) =
            [TERM,CORRECT_ANSWER,ANSWERS].


/* QUALITATIVE RELATIONSHIPS */

if    pick(TERMS) = TERM and
      answers_qualitative_relationships(qualitative_relationships) =
            ANSWERS and
      answer_for_qualitative_relationships(SECTION,TERM) =
            CORRECT_ANSWER and
      left_hand_side_for-SECTION = LEFT_HAND_SIDE and
      display([nl,nl,'I''m still working.',nl,nl])
then  single_question(SECTION,TERMS,qualitative_relationships) =
            [TERM,LEFT_HAND_SIDE,CORRECT_ANSWER,ANSWERS].


/* QUANTITATIVE RELATIONSHIPS */

if    pick(TERMS) = TERM and
      list_of_numbers-1 = CHOICE_OF_NUMBERS and
      pick(CHOICE_OF_NUMBERS) = CHANGE_TERM_VALUE and
      answer_for_quantitative_relationship(SECTION,TERM,
            CHANGE_TERM_VALUE) = [CORRECT_ANSWER,ANSWERS] and
      left_hand_side_for-SECTION = LEFT_HAND_SIDE and
      display([nl,nl,'I''m still working.',nl,nl])
then  single_question(SECTION,TERMS,quantitative_relationships) =
            [TERM,LEFT_HAND_SIDE,CHANGE_TERM_VALUE,
            CORRECT_ANSWER,ANSWERS].


/* TERMS */

if    multipick(TERMS,4) = LIST_OF_TERMS and
      answers_terms(LIST_OF_TERMS) = ANSWERS and
      pick(LIST_OF_TERMS) = TERM and
      answer_for_term(TERM) = CORRECT_ANSWER and
      display([nl,nl,'I''m still working.',nl,nl])
then  single_question(SECTION,TERMS,terms) =
                  [TERM,CORRECT_ANSWER,ANSWERS].
```

```
/*          SINGLE-SECTION_TYPE-OF-SECTION-OF-LESSON          */

nocache(single-section_type-of-SECTION-of-LESSON is complete).

if    slides(LESSON,SECTION) = LIST_OF_SLIDES and
      each-slide_type-of-LIST_OF_SLIDES-of-SECTION is complete
then single-section_type-of-SECTION-of-LESSON is complete.


/*          SINGLE-SLIDE_TYPE-OF-SLIDE-OF-SECTION          */

nocache(single-slide_type-of-SLIDE-of-SECTION is complete).

if    symbols(SECTION,SLIDE) = TERMS and
      material(SECTION,SLIDE) is presented and
      message(patience,[]) and
      quiz_student(SECTION,TERMS,SLIDE) is sought
then single-slide_type-of-SLIDE-of-SECTION is complete.

if    material(SECTION,SLIDE) is presented
then single-slide_type-of-SLIDE-of-SECTION is complete.


/*                       SYMBOLS                       */

nocache(symbols(SECTION,SLIDE)).
noautomaticquestion(symbols(SECTION,SLIDE)).

if    symbols_for-SECTION(terms) = TERMS
then symbols(SECTION,definition1) = TERMS.

if    symbols_for-SECTION(equations) = TERMS
then symbols(SECTION,qualitative_relationships) = TERMS.

if    symbols_for-SECTION(equations) = TERMS
then symbols(SECTION,quantitative_relationships) = TERMS.

if    symbols_for-SECTION(terms) = TERMS
then symbols(SECTION,terms) = TERMS.
```

```
/*                    SYMBOLS_FOR-SECTION                    */

nocache(symbols_for-SECTION(SLIDE)).

symbols_for-equation_1(equations) = ['P(t)',
                                     'R',
                                     'G',
                                     'Sigma',
                                     'Lambda'].

symbols_for-equation_2(equations) = ['P(r)',
                                     'P(t)',
                                     'G',
                                     'Sigma',
                                     'Lambda'].

symbols_for-equation_3(equations) = ['P(r)',
                                     'P(t)',
                                     'A(e)',
                                     't(ot)',
                                     'Sigma',
                                     'Lambda'].

symbols_for-equation_1(terms) = ['P(r)',
                                 'P(t)',
                                 'R',
                                 'G',
                                 'Sigma',
                                 'Lambda'].

symbols_for-equation_3(terms) = ['P(r)',
                                 'P(t)',
                                 'R',
                                 'A(e)',
                                 't(ot)',
                                 'Sigma',
                                 'Lambda'].


/*                      TERM_FOR_ANSWER                      */

nocache(term_for_answer(X)).

term_for_answer(range) = 'R'.
term_for_answer(power_transmitted) = 'P(t)'.
term_for_answer(power_received) = 'P(r)'.
term_for_answer(gain) = 'G'.
term_for_answer(effective_antenna_area) = 'A(e)'.
term_for_answer(time_on_target) = 't(ot)'.
term_for_answer(radar_cross_section) = 'Sigma'.
term_for_answer(wavelength) = 'Lambda'.
```

```
/* ================================================================ */
/* ================================================================ */
/*                    QUIZZING KNOWLEDGE BASE                        */
/* ================================================================ */
/* ================================================================ */


/*                        STUDENT_MODEL-X                           */

/* This is part of the QUIZZING-KB. */


/* DEFINITIONS */

question(student_model-definitions)= [nl,nl,
'Can you define these terms?
     1. No.
     2. I''m not sure.
     3. Yes.'].


/* EQUATIONS */

question(student_model-equations)= [nl,nl,
'Are you familiar with the radar range equation?
     1. I don''t know what it is.
     2. I''ve heard of it.
     3. I''ve used it.'].


/* QUALITATIVE RELATIONSHIPS */

if    student_model-equations = PROFICIENCY
then  student_model-qualitative_relationships = PROFICIENCY.


/* QUANTITATIVE RELATIONSHPS */

if    student_model-equations = PROFICIENCY
then  student_model-quantitative_relationships = PROFICIENCY.


/* TERMS */

question(student_model-terms)= [nl,nl,
'How familiar are you with the terms, such as effective antenna
area and wavelength, and the symbols associated with them?
     1. I''m not familiar with them.
     2. I''m comfortable with them.
     3. I''m very familiar with them.'].

legalvals(student_model-AREA) = [basic,average,advanced].
enumeratedanswers(student_model-AREA).
```

```
/* ================================================================ */
/* ================================================================ */
/*                   STUDENT MODEL KNOWLEDGE BASE                   */
/* ================================================================ */
/* ================================================================ */


/*                       INITIAL_STUDENT_MODEL                     */

nocache(initial_student_model).

if      message(explain_student_model,[]) and
        student_model-terms is sought and
        student_model-definitions is sought and
        student_model-equations is sought
then initial_student_model.
```

```
nocache(student_model_updated(SLIDE)).

if    student_model-SLIDE = average and
      number_of_questions(SLIDE,average) = QUESTIONS_ASKED and
      number_of_correct_answers(SLIDE,QUESTIONS_ASKED) =
            QUESTIONS_CORRECT and
      score(QUESTIONS_ASKED,QUESTIONS_CORRECT) = low and
      do(set student_model-SLIDE = basic)
then  student_model_updated(SLIDE).

if    student_model-SLIDE = advanced and
      number_of_questions(SLIDE,advanced) = QUESTIONS_ASKED and
      number_of_correct_answers(SLIDE,QUESTIONS_ASKED) =
            QUESTIONS_CORRECT and
      score(QUESTIONS_ASKED,QUESTIONS_CORRECT) = low and
      do(set student_model-SLIDE = average)
then  student_model_updated(SLIDE).

if    student_model-SLIDE = basic and
      number_of_questions(SLIDE,basic) = QUESTIONS_ASKED and
      number_of_correct_answers(SLIDE,QUESTIONS_ASKED) =
            QUESTIONS_CORRECT and
      score(QUESTIONS_ASKED,QUESTIONS_CORRECT) = high and
      do(set student_model-SLIDE = average)
then  student_model_updated(SLIDE).

if    student_model-SLIDE = average and
      number_of_questions(SLIDE,average) = QUESTIONS_ASKED and
      number_of_correct_answers(SLIDE,QUESTIONS_ASKED) =
            QUESTIONS_CORRECT and
      score(QUESTIONS_ASKED,QUESTIONS_CORRECT) = high and
      do(set student_model-SLIDE = advanced)
then  student_model_updated(SLIDE).

student_model_updated(SLIDE).
```

```
/* =============================================================== */
/* =============================================================== */
/*                    TUTOR KNOWLEDGE BASE                         */
/* =============================================================== */
/* =============================================================== */


/*                         TUTOR                              */

nocache(tutor(APPROACH,PROFICIENCY,TUTOR_CYCLE)).


/* QUALITATIVE RELATIONSHIP */

    /* BASIC */

if    message(in_denominator_basic_for_qualitative_relationship,
            TERM)
then tutor(TERM-in_denominator-for_qualitative_relationship,
            basic,N).

if    message(in_numerator_basic_for_qualitative_relationship,
            TERM)
then tutor(TERM-in_numerator-for_qualitative_relationship,
            basic,N).

    /* AVERAGE */

if    message(in_denominator_average_for_qualitative_relationship,
            TERM)
then tutor(TERM-in_denominator-for_qualitative_relationship,
            average,N).

if    message(in_numerator_average_for_qualitative_relationship,
            TERM)
then tutor(TERM-in_numerator-for_qualitative_relationship,
            average,N).
```
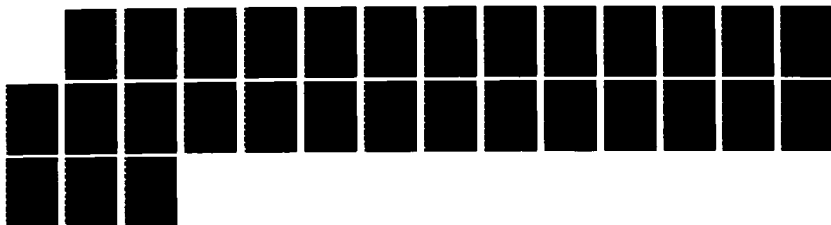
AD-A172 497   AN EXPERT SYSTEM FOR TUTORING INTELLIGENCE ANALYSTS(U)   2/2
              AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL
              OF ENGINEERING R O MELVIN JUN 86 AFIT/GST/ENSS/86M-14
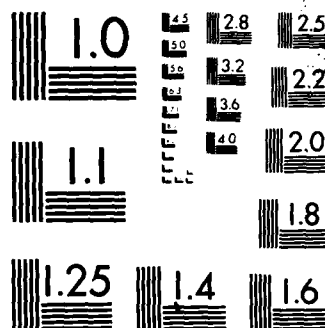UNCLASSIFIED   F/G 5/9   NL

MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

```
/* QUANTITATIVE RELATIONSHIP */

     /* BASIC */

if     message(both_wrong_basic_for_quantitative_relationship,
          [TERM,RELATED,LEFT_HAND_SIDE,FACTOR])
then tutor(TERM-is-RELATED-to-LEFT_HAND_SIDE-by-FACTOR-
          for_quantitative_relationship,basic,N).

if     message(factor_basic_for_quantitative_relationship,
          [TERM,FACTOR,LEFT_HAND_SIDE])
then tutor(FACTOR-for-TERM-for_quantitative_relationship_with-
          LEFT_HAND_SIDE,basic,N).

if     message(in_denominator_basic_for_quantitative_relationship,
          [TERM,LEFT_HAND_SIDE])
then tutor(TERM-in_denominator-for_quantitative_relationship_with-
          LEFT_HAND_SIDE,basic,N).

if     message(in_numerator_basic_for_quantitative_relationship,
          [TERM,LEFT_HAND_SIDE])
then tutor(TERM-in_numerator-for_quantitative_relationship_with-
          LEFT_HAND_SIDE,basic,N).

     /* AVERAGE */

if     message(both_wrong_average_for_quantitative_relationship,
          [TERM,RELATED,LEFT_HAND_SIDE,FACTOR])
then tutor(TERM-is-RELATED-to-LEFT_HAND_SIDE-by-FACTOR-
          for_quantitative_relationship,average,N).

if     message(factor_average_for_quantitative_relationship,
          [TERM,FACTOR,LEFT_HAND_SIDE])
then tutor(FACTOR-for-TERM-for_quantitative_relationship_with-
          LEFT_HAND_SIDE,average,N).

if     message(in_denominator_average_for_quantitative_relationship,
          [TERM,LEFT_HAND_SIDE])
then tutor(TERM-in_denominator-for_quantitative_relationship_with-
          LEFT_HAND_SIDE,average,N).

if     message(in_numerator_average_for_quantitative_relationship,
          [TERM,LEFT_HAND_SIDE])
then tutor(TERM-in_numerator-for_quantitative_relationship_with-
          LEFT_HAND_SIDE,average,N).

if     message(unrelated,TERM)
then tutor(TERM-unrelated,ANY_PROFICIENCY,N).
```

```
/* TERM */

if    message(define_basic,STUDENT_ANSWER)
then tutor(define-STUDENT_ANSWER,basic,N).

if    message(define_average,STUDENT_ANSWER)
then tutor(define-STUDENT_ANSWER,average,N).


/* ADVANCED STUDENT */

if    message(try_again,[])
then tutor(ANY_SLIDE,advanced,N).


/*                 TUTOR-CYCLE-N-OF-(T,C,S,SL)                 */

nocache(tutor-cycle-N-of-(W,X,Y,Z) is complete).
nocache(tutor-cycle-N-of-(V,W,X,Y,Z) is complete).
nocache(tutor-cycle-N-of-(U,V,W,X,Y,Z) is complete).


/* DEFINITIONS */

if    N >= 3 and
      message(tell_answer,CORRECT_ANSWER)
then tutor-cycle-N-of-(ANY_TERM,CORRECT_ANSWER,ANY_STUDENT_ANSWER,
          definitions) is complete.

if    diagnosis(TERM,CORRECT_ANSWER,STUDENT_ANSWER,definitions,N) =
          PROBLEM and
      student_model-definitions = PROFICIENCY and
      metatutor(PROBLEM,STUDENT_ANSWER,N) = APPROACH and
      tutor(APPROACH,PROFICIENCY,N) and
      N + 1 = M and
      number_of_wrong_answers(M) is sought and
      new_answer(definitions,M,[TERM]) = NEW_STUDENT_ANSWER and
      check_answer(CORRECT_ANSWER,NEW_STUDENT_ANSWER,definitions)
          is satisfactory
then tutor-cycle-N-of-(TERM,CORRECT_ANSWER,STUDENT_ANSWER,
          definitions) is complete.


if    N < 3 and
      N + 1 = M and
      new_answer(definitions,M,[TERM]) = NEW_STUDENT_ANSWER and
      tutor-cycle-M-of-(TERM,CORRECT_ANSWER,NEW_STUDENT_ANSWER,
          definitions) is complete
then tutor-cycle-N-of-(TERM,CORRECT_ANSWER,STUDENT_ANSWER,
          definitions) is complete.
```

```
/* QUALITATIVE RELATIONSHIPS */

if    N >= 2 and
      message(tell_answer,CORRECT_ANSWER)
then  tutor-cycle-N-of-(ANY_TERM,LEFT_HAND_SIDE,CORRECT_ANSWER,
          ANY_STUDENT_ANSWER,qualitative_relationships) is
          complete.


if    diagnosis(TERM,CORRECT_ANSWER,STUDENT_ANSWER,
          qualitative_relationships,N) = PROBLEM and
      student_model-qualitative_relationships = PROFICIENCY and
      metatutor(TERM,PROBLEM,CORRECT_ANSWER,N) = APPROACH and
      tutor(APPROACH,PROFICIENCY,N) and
      N + 1 = M and
      number_of_wrong_answers(M) is sought and
      new_answer(qualitative_relationships,M,[TERM,LEFT_HAND_SIDE]) =
          NEW_STUDENT_ANSWER and
      check_answer(CORRECT_ANSWER,NEW_STUDENT_ANSWER,
          qualitative_relationships) is satisfactory
then  tutor-cycle-N-of-(TERM,LEFT_HAND_SIDE,CORRECT_ANSWER,
          STUDENT_ANSWER,qualitative_relationships) is complete.


if    N < 2 and
      N + 1 = M and
      new_answer(qualitative_relationships,M,[TERM,LEFT_HAND_SIDE]) =
          NEW_STUDENT_ANSWER and
      tutor-cycle-M-of-(TERM,LEFT_HAND_SIDE,CORRECT_ANSWER,
          NEW_STUDENT_ANSWER, qualitative_relationships) is complete
then  tutor-cycle-N-of-(TERM,LEFT_HAND_SIDE,CORRECT_ANSWER,
          STUDENT_ANSWER,qualitative_relationships) is complete.
```

```
/* QUANTITATIVE RELATIONSHIPS */

if    N >= 2 and
      CORRECT_ANSWER == [ANSWER,RELATED,EXPONENT] and
      message(tell_answer,ANSWER)
then  tutor-cycle-N-of-(ANY_TERM,LEFT_HAND_SIDE,CHANGE_TERM_VALUE,
          CORRECT_ANSWER,ANY_STUDENT_ANSWER,ORDERED_ANSWERS,
          quantitative_relationships) is complete.

if    diagnosis(TERM,CORRECT_ANSWER,STUDENT_ANSWER,
          quantitative_relationships,N) = PROBLEM and
      student_model-quantitative_relationships = PROFICIENCY and
      metatutor(TERM,PROBLEM,LEFT_HAND_SIDE,CORRECT_ANSWER,N) =
          APPROACH and
      tutor(APPROACH,PROFICIENCY,N) and
      N + 1 = M and
      number_of_wrong_answers(M) is sought and
      new_answer(quantitative_relationships,M,[TERM,LEFT_HAND_SIDE,
          CHANGE_TERM_VALUE,ORDERED_ANSWERS]) = NEW_STUDENT_ANSWER
          and
      check_answer(CORRECT_ANSWER,NEW_STUDENT_ANSWER,
          quantitative_relationships) is satisfactory
then  tutor-cycle-N-of-(TERM,LEFT_HAND_SIDE,CHANGE_TERM_VALUE,
          CORRECT_ANSWER,STUDENT_ANSWER,ORDERED_ANSWERS,
          quantitative_relationships) is complete.

if    N < 2 and
      N + 1 = M and
      new_answer(quantitative_relationships,M,[TERM,LEFT_HAND_SIDE,
          CHANGE_TERM_VALUE,ORDERED_ANSWERS]) = NEW_STUDENT_ANSWER
          and
      tutor-cycle-M-of-(TERM,LEFT_HAND_SIDE,CHANGE_TERM_VALUE,
          CORRECT_ANSWER,NEW_STUDENT_ANSWER,ORDERED_ANSWERS,
          quantitative_relationships) is complete
then  tutor-cycle-N-of-(TERM,LEFT_HAND_SIDE,CHANGE_TERM_VALUE,
          CORRECT_ANSWER,STUDENT_ANSWER,ORDERED_ANSWERS,
          quantitative_relationships) is complete.
```

```
/* TERMS */

if    N >= 3 and
      message(tell_answer,CORRECT_ANSWER)
then tutor-cycle-N-of-(ANY_TERM,CORRECT_ANSWER,ANY_STUDENT_ANSWER,
           terms) is complete.

if    diagnosis(TERM,CORRECT_ANSWER,STUDENT_ANSWER,terms,N) =
           PROBLEM and
      student_model-terms = PROFICIENCY and
      metatutor(PROBLEM,STUDENT_ANSWER,N) = APPROACH and
      tutor(APPROACH,PROFICIENCY,N) and
      N + 1 = M and
      number_of_wrong_answers(M) is sought and
      new_answer(terms,M,[TERM]) = NEW_STUDENT_ANSWER and
      check_answer(CORRECT_ANSWER,NEW_STUDENT_ANSWER,terms)
           is satisfactory
then tutor-cycle-N-of-(TERM,CORRECT_ANSWER,STUDENT_ANSWER,
                         terms) is complete.


if    N < 3 and
      N + 1 = M and
      new_answer(terms,M,[TERM]) = NEW_STUDENT_ANSWER and
      tutor-cycle-M-of-(TERM,CORRECT_ANSWER,NEW_STUDENT_ANSWER,
                         terms) is complete
then tutor-cycle-N-of-(TERM,CORRECT_ANSWER,STUDENT_ANSWER,
                         terms) is complete.


/*                       TUTORING                       */

nocache(tutoring(W,X,Y,Z) is satisfactory).
nocache(tutoring(V,W,X,Y,Z) is satisfactory).
nocache(tutoring(U,V,W,X,Y,Z) is satisfactory).


/* QUALITATIVE RELATIONSHIPS */

if    tutor-cycle-1-of-(TERM,LEFT_HAND_SIDE,CORRECT_ANSWER,
           STUDENT_ANSWER,qualitative_relationships) is complete
then tutoring(TERM,LEFT_HAND_SIDE,CORRECT_ANSWER,STUDENT_ANSWER,
           qualitative_relationships) is satisfactory.


/* QUANTITATIVE RELATIONSHIPS */

if    tutor-cycle-1-of-(TERM,LEFT_HAND_SIDE,CHANGE_TERM_VALUE,
           CORRECT_ANSWER,STUDENT_ANSWER,ORDERED_ANSWERS,
           quantitative_relationships) is complete
then tutoring(TERM,LEFT_HAND_SIDE,CHANGE_TERM_VALUE,CORRECT_ANSWER,
           STUDENT_ANSWER,ORDERED_ANSWERS,
           quantitative_relationships) is satisfactory.
```

```
/* TERMS */

if   tutor-cycle-1-of-(TERM,CORRECT_ANSWER,STUDENT_ANSWER,
        terms) is complete
then tutoring(TERM,CORRECT_ANSWER,STUDENT_ANSWER,
        terms) is satisfactory.
```

```
/* ================================================================ */
/* ================================================================ */
/*              MISCELLANEOUS FUNCTIONS KNOWLEDGE BASE              */
/* ================================================================ */
/* ================================================================ */


/*                          ADVANCE                          */

nocache(advance).

question(advance) = [nl,nl, 'Please type ''c'' to continue.'].
legalvals(advance) = [c].

/*                          COMBINE                          */

nocache(combine(W,X,Y,Z)).

combine(product,ANY_EXPRESSION,SECOND_OPERATOR,SECOND_VALUE) =
        [SECOND_OPERATOR,SECOND_VALUE].

combine(sum,ANY_EXPRESSION,SECOND_OPERATOR,SECOND_VALUE) =
        [SECOND_OPERATOR,SECOND_VALUE].

combine(difference,ANY_EXPRESSION,SECOND_OPERATOR,SECOND_VALUE) =
        [SECOND_OPERATOR,SECOND_VALUE].

if    FIRST_OPERATOR == SECOND_OPERATOR and
      FIRST_VALUE * SECOND_VALUE = NEW_VALUE
then  combine(FIRST_OPERATOR,FIRST_VALUE,SECOND_OPERATOR,
        SECOND_VALUE) = [FIRST_OPERATOR,NEW_VALUE].

if    EXPONENT_VALUE =< ROOT_VALUE and
      EXPONENT_VALUE / ROOT_VALUE = NEW_VALUE
then  combine(exponent,EXPONENT_VALUE,root,ROOT_VALUE) =
        [exponent,NEW_VALUE].

if    EXPONENT_VALUE >= ROOT_VALUE and
      EXPONENT_VALUE / ROOT_VALUE = NEW_VALUE
then  combine(root,ROOT_VALUE,exponent,EXPONENT_VALUE) =
        [exponent,NEW_VALUE].

if    EXPONENT_VALUE >= ROOT_VALUE and
      EXPONENT_VALUE / ROOT_VALUE = NEW_VALUE
then  combine(exponent,EXPONENT_VALUE,root,ROOT_VALUE) =
        [exponent,NEW_VALUE].

if    EXPONENT_VALUE =< ROOT_VALUE and
      EXPONENT_VALUE / ROOT_VALUE = NEW_VALUE
then  combine(root,ROOT_VALUE,exponent,EXPONENT_VALUE) =
        [exponent,NEW_VALUE].
```

B - 34

```
/*                          CONTAINS                          */

noautomaticquestion(contains(X,Y)).
nocache(contains(X,Y)).

contains([exponent,TERM,EXPONENT],TERM) = [exponent,EXPONENT].
contains([root,TERM,ROOT]) = [root,ROOT].
contains([OPERATOR,TERM,ANY_EXPRESSION],TERM) = [exponent,1].
contains([OPERATOR,ANY_EXPRESSION,TERM],TERM) = [exponent,1].

if   LIST == [HEAD:TAIL] and
     contains(LIST,TERM) = [SECOND_OPERATOR,SECOND_VALUE] and
     combine(FIRST_OPERATOR,FIRST_VALUE,SECOND_OPERATOR,
          SECOND_VALUE) = NEW_VALUES
then contains([FIRST_OPERATOR,LIST,FIRST_VALUE],TERM) =
          NEW_VALUES.

if   contains(REST_OF_EXPRESSION,TERM) = VALUES
then contains([OPERATOR,ANY_EXPRESSION,REST_OF_EXPRESSION],TERM) =
          VALUES.


/*                       CONTAINS_QUAL                         */

noautomaticquestion(contains_qual(X,Y)).
nocache(contains_qual(X,Y)).

contains_qual([TERM:TAIL],TERM).
contains_qual([OPERATOR,TERM,ANY_EXPRESSION],TERM).
contains_qual([OPERATOR,ANY_EXPRESSION,TERM],TERM).

if   contains_qual([HEAD:TAIL],TERM)
then contains_qual([OPERATOR,[HEAD:TAIL],ANY_EXPRESSION],TERM).

if   contains_qual([HEAD:TAIL],TERM)
then contains_qual([OPERATOR,ANY_EXPRESSION,[HEAD:TAIL]],TERM).


/*                     CONVERT_RELATIONSHIP                    */

nocache(convert_relationship(X)).

convert_relationship(direct) = inverse.
convert_relationship(inverse) = direct.
```

```
/*                  CONVERT_ROOT_TO_EXPONENT                   */

nocache(convert_root_to_exponent(X,Y)).

if    ROOT_EXPONENT == root and
      1 / ROOT_EXPONENT_VALUE = EXPONENT_VALUE
then convert_root_to_exponent(ROOT_EXPONENT,
          ROOT_EXPONENT_VALUE) = EXPONENT_VALUE.

if    not(ROOT_EXPONENT == root)
then convert_root_to_exponent(ROOT_EXPONENT,
          ROOT_EXPONENT_VALUE) = ROOT_EXPONENT_VALUE.


/*                  EACH-TYPE-OF-ELEMENT-OF-WHOLE              */

nocache(each-TYPE-of-ELEMENT-of-WHOLE is complete).

each-TYPE-of-[]-of-WHOLE is complete.

if    single-TYPE-of-ELEMENT-of-WHOLE is complete and
      each-TYPE-of-TAIL-of-WHOLE is complete
then each-TYPE-of-[ELEMENT:TAIL]-of-WHOLE is complete.

if    single-TYPE-of-ELEMENT-of-WHOLE is complete
then each-TYPE-of-ELEMENT-of-WHOLE is complete.


/*                  FINISHED_WITH_LESSON                       */

if    display([nl,
'I hope this has been helpful.' ,nl])
then finished_with_lesson.


/*                        LENGTH                               */

nocache(length(X)).

length([]) = 0.
length([A]) = 1.
length([A,B]) = 2.
length([A,B,C]) = 3.
length([A,B,C,D]) = 4.

if    length(TAIL) = M and
      M + 4 = N
then length([A,B,C,D:TAIL]) = N.
```

```
/*                      MATERIAL                         */

nocache(material(X,Y) is presented).

if   presentation(SECTION,SLIDE) = P and
     display(P) and
     advance is sought
then material(SECTION,SLIDE) is presented.


/*                      MESSAGE                          */

nocache(message(ID,LIST_OF_ARGUMENTS)).

if   term_for_answer(STUDENT_ANSWER) = A and
     display([nl,
'No, ',STUDENT_ANSWER,' is represented by ',A,'.  Remember
that with many terms, you can use a neumonic device.',nl])
then message(define_basic,STUDENT_ANSWER).

if   term_for_answer(STUDENT_ANSWER) = A and
     display([nl,
'No, ',STUDENT_ANSWER,' is represented by ',A,'.',nl])
then message(define_average,STUDENT_ANSWER).


if   message_text(ID,LIST_OF_ARGUMENTS) = TEXT_OF_MESSAGE and
     display(TEXT_OF_MESSAGE)
then message(ID,LIST_OF_ARGUMENTS).


/*                      MESSAGE_TEXT                     */

nocache(message_text(ANYID,ANY_LIST_OF_ARGUMENTS)).

message_text(confirm_answer,CORRECT_ANSWER) = [nl,
'Your answer of ' ,CORRECT_ANSWER, ' is correct.' ,nl].

message_text(explain_student_model,[]) = [nl,
'I will ask you a couple of questions to help me better direct
this lesson.'].

message_text(patience,[]) = [nl,
'Please be patient while I make up some questions.' ,nl].

message_text(tell_answer,CORRECT_ANSWER) = [nl,
'The correct answer is ',CORRECT_ANSWER, '.' ,nl].

message_text(try_again,[]) = [nl,
'Your answer is incorrect.',nl].
```

```
/* QUALITATIVE RELATIONSHIPS */

      /* BASIC */

message_text(in_numerator_basic_for_qualitative_relationship,
          TERM) = [nl,
TERM,' is in the numerator which means it is not inversely
related.',nl].

message_text(in_denominator_basic_for_qualitative_relationship,
          TERM) = [nl,
TERM, ' is in the denominator which means it is not directly
related.',nl].

      /* AVERAGE */

message_text(in_numerator_average_for_qualitative_relationship,
          TERM) = [nl,
'No, ',TERM, ' is in the numerator.',nl].

message_text(in_denominator_average_for_qualitative_relationship,
          TERM) = [nl,
'No, ',TERM, ' is in the denominator.',nl].

message_text(unrelated,TERM) = [nl,
'No, ',TERM, ' is not in this equation.',nl].


/* QUANTITATIVE RELATIONSHIPS */
```

```
        /* BASIC */

message_text(in_numerator_basic_for_quantitative_relationship,
            [TERM,LEFT_HAND_SIDE]) = [nl,
'You have the correct power but the wrong relationship between ',nl,
LEFT_HAND_SIDE,' and ',TERM,'.  ',TERM,' is in the numerator and
directly related to ',LEFT_HAND_SIDE, '.',nl].

message_text(in_denominator_basic_for_quantitative_relationship,
            [TERM,LEFT_HAND_SIDE]) = [nl,
'You have the correct power but the wrong relationship between ',nl,
LEFT_HAND_SIDE,' and ',TERM,'.  ',TERM,' is in the denominator and
inversely related to ',LEFT_HAND_SIDE, '.',nl].

message_text(both_wrong_basic_for_quantitative_relationship,
        [TERM,RELATED,LEFT_HAND_SIDE,FACTOR]) = [nl,
'Both the power and the relationship between ',LEFT_HAND_SIDE,nl,
' and ',TERM,' are incorrect.  'LEFT_HAND_SIDE,' is ',
RELATED,'ly related to ',TERM, ' by a power of ',FACTOR,'.',nl].

message_text(factor_basic_for_quantitative_relationship,
            [TERM,FACTOR,LEFT_HAND_SIDE]) = [nl,
'You have the correct relationship between ',LEFT_HAND_SIDE,' and ',
TERM,' but the incorrect power.  The power is ',FACTOR,'.',nl].


        /* AVERAGE */

message_text(in_numerator_average_for_quantitative_relationship,
            [TERM,LEFT_HAND_SIDE]) = [nl,
'No, ',TERM, ' is in the numerator.',nl].

message_text(in_denominator_average_for_quantitative_relationship,
            [TERM,LEFT_HAND_SIDE]) = [nl,
'No, ',TERM, ' is in the denominator.',nl].

message_text(unrelated,TERM) = [nl,
'No, ',TERM, ' is not in this equation.',nl].

message_text(both_wrong_average_for_quantitative_relationship,
        [TERM,RELATED,LEFT_HAND_SIDE,FACTOR]) = [nl,
'No, ',LEFT_HAND_SIDE, ' is ',RELATED, 'ly related to ',TERM,' by
a power of ', FACTOR, '.',nl].

message_text(factor_average_for_quantitative_relationship,
            [TERM,FACTOR,LEFT_HAND_SIDE]) = [nl,
'No, the factor for ',TERM, ' is ',FACTOR,'.',nl].
```

```
/*                      MULTIPICK                          */

nocache(multipick(X,Y)).

if    length(LIST) = LENGTH and
      multipick1(LIST,NUMBER,LENGTH) = RESULT
then multipick(LIST,NUMBER) = RESULT.


/*                      MULTIPICK1                         */

nocache(multipick1(X,Y,Z)).

multipick1(ANY_LIST,0,ANYNUMBER) = [].

if    NUMBER >= LENGTH
then multipick1(LIST,NUMBER,LENGTH) = LIST.

if    LENGTH - NUMBER = DIFFERENCE and
      DIFFERENCE < NUMBER and
      remv-DIFFERENCE-from-LIST = NEW_LIST
then multipick1(LIST,NUMBER,LENGTH) = NEW_LIST.

if    pick(LIST) = ELEMENT and
      remove(ELEMENT,LIST) = NEW_LIST and
      NUMBER - 1 = NEW_NUMBER and
      LENGTH - 1 = NEW_LENGTH and
      multipick1(NEW_LIST,NEW_NUMBER,NEW_LENGTH) = TAIL
then multipick1(LIST,NUMBER,LENGTH) = [ELEMENT:TAIL].


/*                      NTH_TERM                           */

nocache(nth_term(X,Y)).

nth_term(1,[A:B]) = A.
nth_term(2,[A,B:C]) = B.
nth_term(3,[A,B,C:D]) = C.
nth_term(4,[A,B,C,D:E]) = D.

if    NUMBER - 4 = SMALLER_NUMBER and
      nth_term(SMALLER_NUMBER,REST_OF_LIST) = TERM
then nth_term(NUMBER,[A,B,C,D:REST_OF_LIST]) = TERM.
```

```
/*                    NUMBER_OF_CORRECT_ANSWERS              */

nocache(number_of_correct_answers(SLIDE,QUESTIONS_ASKED_)).

if    number_wrong = QUESTIONS_WRONG and
      QUESTIONS_ASKED - QUESTIONS_WRONG = QUESTIONS_CORRECT and
      do(reset number_wrong)
then number_of_correct_answers(SLIDE,QUESTIONS_ASKED) =
          QUESTIONS_CORRECT.

number_of_correct_answers(SLIDE,QUESTIONS_ASKED) =
          QUESTIONS_ASKED.


/*                    NUMBER_OF_WRONG_ANSWERS                */

nocache(number_of_wrong_answers(ANY_NUMBER)).

if    number_wrong is unknown and
      do(set number_wrong = 1)
then number_of_wrong_answers(2).

if    number_wrong = NUMBER_INCORRECT and
      NUMBER_INCORRECT + 1 = UPDATED_NUMBER_INCORRECT and
      do(set number_wrong = UPDATED_NUMBER_INCORRECT)
then number_of_wrong_answers(2).

number_of_wrong_answers(ANYNUMBER).


/*                         NUMBER_WRONG                      */

noautomaticquestion(number_wrong).


/*                            PICK                           */

nocache(pick(X)).

if    length(LIST) = NUMBER and
      random(NUMBER) = RANDOM_NUMBER and
      nth_term(RANDOM_NUMBER,LIST) = TERM
then pick(LIST) = TERM.
```

```
/*                          POWER                          */

nocache(power(CHANGE_TERM_VALUE,POWER_VALUE)).

power(CHANGE_TERM_VALUE,1) = CHANGE_TERM_VALUE.

if    POWER_VALUE - 1 = NEW_POWER_VALUE and
      power(CHANGE_TERM_VALUE,NEW_POWER_VALUE) = RESULT and
      CHANGE_TERM_VALUE * RESULT = RELATIONSHIP
then power(CHANGE_TERM_VALUE,POWER_VALUE) = RELATIONSHIP.


/*                          RANDOM                          */

nocache(random(X)).

if    seed = SEED and
      (SEED mod MAXIMUM_VALUE) + 1 = RANDOM_NUMBER and
      do(reset seed) and
      (125 * SEED + 1) mod 4096 = NEW_SEED and
      do(set seed = NEW_SEED)
then random(MAXIMUM_VALUE) = RANDOM_NUMBER.
```

```
/*                          RELATED                              */

nocache(related(X,Y)).

if    related(EXPRESSION,TERM) = RELATIONSHIP
then  related([root,EXPRESSION,ROOT],TERM) = RELATIONSHIP.

if    related(EXPRESSION,TERM) = RELATIONSHIP
then  related([exponent,EXPRESSION,EXPONENT],TERM) = RELATIONSHIP.

if    contains_qual(NUMERATOR,TERM)
then  related([quotient,NUMERATOR,DENOMINATOR],TERM) = direct.

if    contains_qual(DENOMINATOR,TERM)
then  related([quotient,NUMERATOR,DENOMINATOR],TERM) = inverse.

if    (contains_qual(FACTOR1,TERM) or
       contains_qual(FACTOR2,TERM))
then  related([sum,FACTOR1,FACTOR2],TERM) = direct.

if    (contains_qual(FACTOR1,TERM) or
       contains_qual(FACTOR2,TERM))
then  related([difference,FACTOR1,FACTOR2],TERM) = direct.

if    (contains_qual(FACTOR1,TERM) or
       contains_qual(FACTOR2,TERM))
then  related([product,FACTOR1,FACTOR2],TERM) = direct.

related(EQUATION,TERM) = unrelated.


/*                          REMOVE                               */

nocache(remove(X,Y)).

remove(A,[]) = [].
remove(A,[A:TAIL]) = TAIL.
remove(B,[A,B:TAIL]) = [A:TAIL].
remove(C,[A,B,C:TAIL]) = [A,B:TAIL].
remove(D,[A,B,C,D:TAIL]) = [A,B,C:TAIL].

if    remove(TERM,TERMS_IN_LIST) = NEW_LIST
then  remove(TERM,[A,B,C,D:TERMS_IN_LIST]) = [A,B,C,D:NEW_LIST].
```

```
nocache(remv-NUMBER-from-LIST).

if    pick(LIST) = ELEMENT and
      remove(ELEMENT,LIST) = NEW_LIST
then  remv-1-from-LIST = NEW_LIST.

if    NUMBER - 1 = NEW_NUMBER and
      remv-NEW_NUMBER-from-LIST = LIST1 and
      pick(LIST1) = ELEMENT and
      remove(ELEMENT,LIST1) = RESULT
then  remv-NUMBER-from-LIST = RESULT.
```

/*   ROOT   */

```
nocache(root(CHANGE_TERM_VALUE,ROOT_VALUE)).

root(CHANGE_TERM_VALUE,1.0) = CHANGE_TERM_VALUE.

if    float(CHANGE_TERM_VALUE) = REAL_CHANGE_TERM_VALUE and
      ROOT_VALUE * 2 = NEW_ROOT_VALUE and
      root(REAL_CHANGE_TERM_VALUE,NEW_ROOT_VALUE) =
            INTERMEDIATE_RESULT and
      sqrt(INTERMEDIATE_RESULT) = CHANGE
then  root(CHANGE_TERM_VALUE,ROOT_VALUE) = CHANGE.
```

/*   ROUND_NUMBER   */

```
nocache(round_number(X,Y)).

if    power(10,PLACES) = FACTOR and
      real_round(NUMBER * FACTOR) / FACTOR = RESULT
then  round_number(NUMBER,PLACES) = RESULT.
```

```
/*                              SCORE                              */

nocache(score(QUESTIONS_ASKED,QUESTIONS_CORRECT)).

if    QUESTIONS_CORRECT/QUESTIONS_ASKED = PERCENT_CORRECT and
      PERCENT_CORRECT < 0.5
then  score(QUESTIONS_ASKED,QUESTIONS_CORRECT) = low.

if    QUESTIONS_CORRECT/QUESTIONS_ASKED = PERCENT_CORRECT and
      PERCENT_CORRECT >= 0.5 and
      PERCENT_CORRECT =< 0.75
then  score(QUESTIONS_ASKED,QUESTIONS_CORRECT) = medium.

if    QUESTIONS_CORRECT/QUESTIONS_ASKED = PERCENT_CORRECT and
      PERCENT_CORRECT > 0.75
then  score(QUESTIONS_ASKED,QUESTIONS_CORRECT) = high.


/*                              SEED                               */

question(seed) = [nl, 'Please enter the day of the month.  For
instance if this is the May 15, enter ''15''.'].

legalvals(seed) = integer.
```

```
nocache(translate(X,Y,Z)).

if      EXPONENT_VALUE < 1.0 and
        root(CHANGE_TERM_VALUE,EXPONENT_VALUE) = CHANGE and
        round_number(CHANGE,3) = ROUNDED_CHANGE
then    translate(CHANGE_TERM_VALUE,direct,EXPONENT_VALUE) =
             [ROUNDED_CHANGE,direct,EXPONENT_VALUE].

if      EXPONENT_VALUE < 1.0 and
        root(CHANGE_TERM_VALUE,EXPONENT_VALUE) = CHANGE and
        1/CHANGE = INVERSE_CHANGE and
        round_number(INVERSE_CHANGE,3) = INVERSE_ROUNDED_CHANGE
then    translate(CHANGE_TERM_VALUE,inverse,EXPONENT_VALUE) =
             [INVERSE_ROUNDED_CHANGE,inverse,EXPONENT_VALUE].

if      EXPONENT_VALUE >= 1.0 and
        power(CHANGE_TERM_VALUE,EXPONENT_VALUE) = CHANGE and
        round_number(CHANGE,3) = ROUNDED_CHANGE
then    translate(CHANGE_TERM_VALUE,direct,EXPONENT_VALUE) =
             [ROUNDED_CHANGE,direct,EXPONENT_VALUE].

if      EXPONENT_VALUE >= 1.0 and
        power(CHANGE_TERM_VALUE,EXPONENT_VALUE) = CHANGE and
        1/CHANGE = INVERSE_CHANGE and
        round_number(INVERSE_CHANGE,3) = INVERSE_ROUNDED_CHANGE
then    translate(CHANGE_TERM_VALUE,inverse,EXPONENT_VALUE) =
             [INVERSE_ROUNDED_CHANGE,inverse,EXPONENT_VALUE].
```

```
/* =============================================================== */
/* =============================================================== */
/*                      LISTING OF RULES                           */
/* =============================================================== */
/* =============================================================== */
```

/* Control Rules Knowledge Base

        Lesson
        Lesson_complete

    Diagnostics Knowledge Base

        Diagnosis

    Lesson Knowledge Base

        Presentation

    Metatutor Knowledge Base

        Metatutor

    Presentation Knowledge Base

        Sections
        Slides

Question Generator and Solver Knowledge Base

    Answer_for_definition
    Answer_for_qualitative_relationships
    Answer_for_quantitative_relationship
    Answers_quantitative_relationships
    Answer_for-term
    Answers_definitions
    Answers_qualitative_relationships
    Answers_terms
    Ask_a_question
    Check_answer
    Define_equation
    Different_from_exponent_value
    Left_hand_side_for_equation_x
    List_of_numbers_x
    Make_questions
    New_answer
    Number_of_questions
    Order_answers
    Put_direct_first
    Quantitative
    Quiz_student
    Scramble
    Single_ask_question-of-list-of-slide
    Single_question
    Single_section_type-of-section-of-lesson
    Single-slide_type-of-slide-of-section
    Symbols
    Symbols_for_section
    Term_for_answer

Quizzing Knowledge Base

    Student_model-x

Student Model Knowledge Base

    Initial_student_model
    Student_model_updated

Tutoring Knowledge Base

    Tutor
    Tutor-cycle-n-of-list
    Tutoring

Miscellaneous Functions Knowledge Base

Advance
Combine
Contains
Contains_qual
Convert_relationship
Convert_root_to_exponent
Each-type-of-element-of-whole
Finished_with_lesson
Length
Material
Message
Message_text
Multipick
Multipick1
Nth_term
Number_of_correct_answers
Number_of_wrong_answers
Number_wrong
Pick
Power
Random
Related
Remove
Remv-n-from-list
Root
Round_number
Score
Seed
Translate

Bibliography

Barr, Avron and Edward A. Feigenbaum. The Handbook of
   Artificial Intelligence, Volume 2. Los Altos CA:
   William Kaufmann, Inc., 1982.

Brown, John Seeley et al. "Pedagogical, Natural Language and
   Knowledge Engineering Techniques in SOPHIE I, II, III,"
   Intelligent Tutoring Systems, edited by D. Sleeman and
   J.S. Brown. Orlando: Academic Press, Inc., 1982.

Burton, Richard R. and John Seely Brown. "An Investigation
   of Computer Coaching for Informal Learning Activities,"
   Intelligent Tutoring Systems, edited by D. Sleeman and
   J.S. Brown. Orlando: Academic Press, Inc., 1982.

Burton, Richard R. "Diagnosing Bugs in a Simple Procedural
   Skill," Intelligent Tutoring Systems, edited by
   D. Sleeman and J.S. Brown. Orlando: Academic Press,
   Inc., 1982.

Carbonell, Jaime R. "AI in CAI: An Artificial-Intelligence
   Approach to Computer-Assisted Instruction," IEEE
   Transactions on Man-Machine Systems, MMS-11: 190-202
   (December 1970).

Clancey, William J. "Tutoring Rules for Guiding a Case
   Method Dialogue," Intelligent Tutoring Systems, edited
   by D. Sleeman and J.S. Brown. Orlando: Academic Press,
   Inc., 1982.

Clancey, William J. "Heuristic Classification," Artificial
   Intelligence, 27: 289-350 (December 1985).

Clocksin, William F. and Christopher S. Mellish. Programming
   in Prolog (Second Edition). Berlin: Springer-Verlag,
   1984.

Davis, Randall. "Expert Systems: Where Are We? And Where Do
   We Go From Here?," AI Magazine, 3: 3-22 (Spring 1982).

de Kleer, Johan. "Qualitative and Quantitative Reasoning in
   Classical Mechanics," Artificial Intelligence: An MIT
   Perspective, Volume I, edited by Patrick H. Winston and
   Richard H. Brown. Cambridge: The MIT Press, 1979.

Freedman, Roy S. and Jeffrey P. Rosenking. "Designing
   Computer-Based Training Systems: OBIE-1:KNOBE," IEEE
   Expert, 1: 31-38 (Summer 1986).

Gable, Alice and Carl V. age. "The Use of Artificial
    Intelligence Techniques in Computer-Assisted Instruction:
    an Overview," International Journal of Man-Machine
    Studies, 12: 259-282 (April 1980).

Genesereth, Michael R. "The Role of Plans in Intelligent
    Teaching Systems," Intelligent Tutoring Systems, edited
    by D. Sleeman and J.S. Brown. Orlando: Academic Press,
    Inc., 1982.

King, David and Paul Harmon. Expert Systems. New York:
    John Wiley & Sons, 1985.

Kovacs, William., Technical Group Leader. Personal
    Interviews. Foreign Technology Division,
    Wright-Patterson AFB, OH. September through November
    1985.

M1-D-1004-00. M.1 Reference Manual for Software Version 2.0.
    Teknowledge, Inc., Palo Alto CA, 1986.

M1-D-5003-00. M.1 Sample Knowledge Systems. Teknowledge,
    Inc., Palo Alto CA, 1986.

National Security Agency Central Security Service. Technical
    Elint Analyst Training Material, Volume I. W23/WA-001-81.
    Fort George G. Meade MD, 23 October 1981.

O'Shea, Tim and John Self. Learning and Teaching with
    Computers: Artificial Intelligence in Education.
    Englewood Cliffs NJ: Prentice-Hall Inc., 1983.

Pyati, V.P. Lecture materials d'stributed in EE 573,
    Electronic Warfare. School of Engineering, Air Force
    Institute of Technology (AU), Wright-Patterson AFB OH,
    July through September 1985.

Rich, Elaine. Artificial Intelligence. New York: McGraw-
    Hill, Inc., 1983.

Roberts, Franklin C. and Ok-choon Park. "Intelligent
    Computer-Assisted Instruction: An Explanation and
    Overview," Educational Technology, 23: 7-12 (December
    1983).

Scandura, Joseph M. "Theory-Driven Expert Systems: The Next
    Step in Computer Software," Educational Technology, 24:
    47-48 (November 1984).

Skolnick, Merrill I. Introduction to Radar Systems (Second
    Edition). New York: McGraw-Hill Book Company, 1980.

Stefik, Mark. "The Organization of Expert Systems: A
    Tutorial," Artificial Intelligence, 18: 135-173 (March
    1982).

Stevens, Albert et al. "Misconceptions in Students'
    Understanding," Intelligent Tutoring Systems, edited by
    D. Sleeman and J.S. Brown. Orlando: Academic Press,
    Inc., 1982.

Stimson, George W. Introduction to Airborne Radar. El
    Segundo CA: Hughes Aircraft Company, 1983.

Stine, Gregory., Signal Analyst. Personal interviews.
    Foreign Technology Division, Wright-Patterson AFB, OH.
    January through May 1986.

Stuart, John A. et al. "Assessment of Instructor Performance
    in the Military Service Schools," Educational Technology,
    24: 12-15 (September 1984).

Waterman, Donald A. A Guide to Expert Systems. Reading MA:
    Addison-Wesley Publishing Co., 1986.

Wedman, John F. and Greg P. Stefanich. "Guidelines for
    Computer-Based Testing of Student Learning of Concepts,
    Principles, and Procedures," Educational Technology, 24:
    23-28 (June 1984).

Vita

Major Richard O. Melvin was born on 25 April 1948 in Elgin, Illinois. Upon graduation from Carthage Community High School, Carthage, Illinois in 1966, he attended the University of Illinois and graduated with a Bachelor of Science degree in Agricultural Engineering in June 1971.

A senior pilot, Major Melvin began his flying career flying RC-135's as an electronic warfare officer with the 24th Strategic Reconnaissance Squadron, Eielson AFB, Alaska. While there, he was selected to attend pilot training at Webb AFB, Texas. Upon graduation in July 1976, he flew B-52D's with the 60th Bombardment Squadron, Anderson AFB, Guam. In August 1980, Major Melvin was transferred to the 46th Bombardment Squadron, Grand Forks AFB, North Dakota, where he flew both the B-52G and B-52H. In February 1983, he became the 319th Bombardment Wing Flying Safety Officer at Grand Forks AFB. In August 1984 he entered the School of Engineering, Air Force Institute of Technology.

                    Permanent Address: R. R. 1; Box 138
                                       Carthage, Illinois 62321

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS			
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited			
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GST/ENS/86M-14	5. MONITORING ORGANIZATION REPORT NUMBER(S)			

6a. NAME OF PERFORMING ORGANIZATION School of Engineering	6b. OFFICE SYMBOL (If applicable) AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433		7b. ADDRESS (City, State and ZIP Code)

8a. NAME OF FUNDING/SPONSORING ORGANIZATION Foreign Technology Div	8b. OFFICE SYMBOL (If applicable) FTD/SQS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER			
8c. ADDRESS (City, State and ZIP Code) Wright-Patterson AFB, OH 45433		10. SOURCE OF FUNDING NOS.			
		PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

11. TITLE (Include Security Classification)
See Box 19

12. PERSONAL AUTHOR(S)
Richard O. Melvin, Major, USAF

13a. TYPE OF REPORT MS Thesis	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) 1986 June	15. PAGE COUNT 123

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Computer Aided Instruction, Expert Systems,
09	02		Artificial Intelligence, Training
06	04		

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

Title: An Expert System for Tutoring Intelligence Analysts

Thesis Chairman: Gregory S. Parnell, Major, USAF

Approved for public release: IAW AFR 190-1/.
ANN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AFGI)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Gregory S. Parnell, Major, USAF	22b. TELEPHONE NUMBER (Include Area Code) (513) 255-3362	22c. OFFICE SYMBOL AFIT/ENS

Abstract

Intelligent Computer Aided Instruction (ICAI) allows a computer to perform some of the functions normally performed by a human instructor. This thesis describes the design and implementation of an ICAI system which presents radar principles to a student, tests him, finds out why he made an error, and then corrects the error.

To allow the system to be used for different subject domains, the knowledge required to teach was kept separate from the knowledge about the subject domain. During the design phase, the knowledge about teaching was partitioned into eight knowledge bases, the functions of these knowledge bases were described, and their interactions with one another were shown. During the implementation phase, each knowledge base was developed separately before being integrated into the system. Of these eight knowledge bases, the knowledge base which tests the student received the major emphasis. This knowledge base generates a multiple choice question, finds the answer, and creates plausible incorrect answers to serve as distractors. Although the radar range equation was used to test this knowledge base, this knowledge base performs the same functions with any equation which is entered into the system in its canonical form.

Although this thesis establishes the framework for an ICAI system and then demonstrates its feasibility, further research is required before the system can be used in an actual training environment.

END

11 – 86

DTIC