DTIC FILE COPY

AD-A198 554

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN INTELLIGENT COMPUTER-ASSISTED INSTRUCTION SYSTEM FOR CARDIOPULMONARY RESUSCITATION

by

Debra S. Campbell

June 1988

Thesis Advisor: Neil C. Rowe

Approved for public release; distribution is unlimited

DTIC ELECTE AUG 3 0 1988 D E

88 8 30 042

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 is Unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): Code 52Rp	7a NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code): Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)

AN INTELLIGENT COMPUTER-ASSISTED INSTRUCTION SYSTEM FOR CARDIOULMONARY RESUSCITATION

12 PERSONAL AUTHOR(S)
Campbell, Debra S.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM _____ TO _____	1988 June	145

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Intelligent Computer-Assisted Instruction (ICAI) Computer-Based Instruction (CBI)

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

This study discusses the design and implementation of an Intelligent Computer-Assisted Instruction system for cardioulmonary resuscitation. Utilizing artificial-intelligence techniques, the system combines a learning-while-doing environment with effective guidance of tutorial interactions.

The user's knowledge of CPR procedures is tested at one of three experience levels, utilizing a randomly generated scenario. Using means-ends analysis, the recommended action is determined for each successive state in the scenario. This action is compared with the user's selection. If a difference exists, an hypothesis guides the tutoring module in the selection of a tutoring strategy.

An on-line review of CPR procedures is available, as is a help function to provide direction to the user if needed. At the end of a session, a summary of the user's actions is provided.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL: Prof. Neil C. Rowe	22b TELEPHONE (Include Area Code): (408) 646-2462	22c OFFICE SYMBOL: Code 52Rp

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

☆ U.S. Government Printing Office: 1986—606-24

Approved for public release; distribution is unlimited.

AN INTELLIGENT COMPUTER-ASSISTED INSTRUCTION SYSTEM for CARDIOPULMONARY RESUSCITATION

by

Debra S. Campbell
Lieutenant Commander, United States Navy
B.A., University of Washington, 1976

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

June 1988

Author: Debra S. Campbell

Debra S. Campbell

Approved by: Neil C. Rowe

Neil C. Rowe, Thesis Advisor

Gordon H. Bradley, Second Reader

Robert B. McGhee, Acting Chairman,
Department of Computer Science

James M. Fremgen,
Acting Dean of Information and Policy Sciences

ABSTRACT

This study discusses the design and implementation of an Intelligent Computer-Assisted Instruction system for cardiopulmonary resuscitation. Utilizing artificial-intelligence techniques, the system combines a learning-while-doing environment with effective guidance of tutorial interactions.

The user's knowledge of CPR procedures is tested at one of three experience levels utilizing a randomly generated scenario. Using means-ends analysis, the recommended action is determined for each successive state in the scenario. This action is compared with the user's selection. If a difference exists, an hypothesis is formulated concerning the cause and severity of the difference. This hypothesis guides the tutoring module in the selection of a tutoring strategy.

An on-line review of CPR procedures is available, as is a help function to provide direction to the user if needed. At the end of a session, a summary of the user's actions is provided.

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

DTIC COPY INSPECTED 1

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

I. INTRODUCTION 1
A. CPR 1
B. METHODOLOGY 3
C. PREVIEW 4
II. OVERVIEW OF COMPUTER-ASSISTED INSTRUCTION 6
A. BACKGROUND 6
B. COMPUTER-BASED INSTRUCTION 6
C. ICAI SYSTEMS 7
1. Structural Methodology 8
a. Expert Module 8
b. Tutor Module 8
c. Student Model 10
2. Applications 11
D. COMPARISON AND CONTRAST OF CBI AND ICAI 13
1. Focus 13
2. Subject Matter Domain 14
3. Format 14
4. System Design Structure 15
5. Development Process 16
6. Hardware and Software 16
III. OVERVIEW OF CPR 18
A. PURPOSE 18
B. PROCEDURES 18
C. TRAINING PROGRAMS 24
IV. CPR TRAINING: AN ICAI APPROACH 26
A. SYSTEMS ORGANIZATION 26
B. EXPERT MODULE 26
1. Inference Engine 26
2. Knowledge Base 28
C. KNOWLEDGE ACQUISITION 34
D. TUTOR MODULE 37
E. STUDENT MODEL 39
F. USER INTERFACE 40
1. Input 40
a. Menu Advantages 40

b. Menu Disadvantages 40
c. Application 41
2. Output 45
V. SYSTEM USE 47
A. RESOURCE REQUIREMENTS 47
1. System Requirements 47
2. Program Language 47
B. DEMONSTRATION 47
C. LIMITATIONS 48
1. Realistic Simulation 48
2. Accuracy 48
a. Parallelism 48
b. Careless Errors 48
D. RECOMMENDED SYSTEM ENHANCEMENTS 49
1. Operator Selection 49
2. Student Modeling 49
3. Biased Randomness 50
4. Graphics 50
5. Portability 50
VI. SUMMARY 51
APPENDIX A - NOVICE LEVEL DEMONSTRATION 52
APPENDIX B - INTERMEDIATE LEVEL DEMONSTRATION 68
APPENDIX C - ADVANCED LEVEL DEMONSTRATION 84
APPENDIX D - SOURCE CODE 104
APPENDIX E - ROWE SOURCE CODE 122
LIST OF REFERENCES 134
INITIAL DISTRIBUTION LIST 135

LIST OF FIGURES

Figure 2.1	ICAI System Components	9
Figure 3.1	CPR Procedural Flowchart - Part One	20
Figure 3.2	CPR Procedural Flowchart - Part Two	21
Figure 4.1	CPR Tutor Structural Model	27
Figure 4.2	CPR Tutor Sample Preconditions	30
Figure 4.3	CPR Tutor Sample Postconditions	32
Figure 4.4	CPR Tutor Random Substitutions	33
Figure 4.5	CPR Tutor Main Menu	41
Figure 4.6	Novice User Menu	42
Figure 4.7	Intermediate User Menu	43
Figure 4.8	Advanced User Menu	44
Figure 4.9	Repetition Qualifier Menu for the Advanced User	45

ACKNOWLEDGEMENTS

The author wishes to express her gratitude to a number of people who supported this work. To my advisor, **Professor Neil C. Rowe**, who assisted me with the initial idea and direction to start the project, and then stepped back, allowing me the freedom to learn through exploration. This research could not have been completed without his support. His provision of the original version of the *metutor* and the inclusive subroutines enabled this project to go into more depth than could ever have been done otherwise in the limited amount of time available. To **Professor Gordon Bradley** for agreeing "sight unseen" to take the time and effort necessary to be my second reader. Finally, I would like to personally thank my husband, **Fred**, for the understanding, support and patience he provided.

I. INTRODUCTION

Computer-based instruction can dramatically alter the way we learn and teach. A significant advantage of using the computer in the realm of education and training is that it allows for interactive learning for all students. The needs of each student can be met by individualizing the learning experience. In the typical classroom environment there are relatively few opportunities for such interaction between the teacher and the student. Private human tutors can provide this one-on-one interaction; however, they can be cost-prohibitive. The human tutor is also limited in the knowledge they can possess. Computerized instructors, however, can capture the knowledge of many experts and can provide interactive, individualized instruction.

With such enormous potential, one might ask why computers have not had more of an impact in the world of education. Computer-based instruction has been available for many years. The impact of this instructional media, however, has been small. There are many possible reasons for this lack of positive impact. One answer lies in the ability to think. The processes of teaching and learning are fundamentally cognitive activities. To successfully teach, computers must be able to capture the skill of "thinking".

The intent of this thesis is to explore the feasibility of instruction via a "thinking" computer utilizing cardiopulmonary resuscitation (CPR) as the application domain. The reason for this choice of application is its real life applicability to a wide range of students.

A. CPR

The leading cause of death in the United States is cardiovascular disease. It is estimated that a million and a half Americans will have heart attacks each year and one

third of these people will die. This equates to a death rate of 1500 people every day from heart attacks alone. In addition to heart attacks, victims of other situations such as drowning accidents, electrocution, drug overdose and poisoning may also go into cardiac arrest. Most of these victims who die from a cardiovascular failure do so before they ever reach trained medical personnel and emergency equipment. It is therefore the chance bystander who can play a critical role in reducing this death rate by being able to recognize cardiac emergencies, provide proper first aid, and call for medical services.

If a person's heart stops for any reason, cardiopulmonary resuscitation (CPR) must be started immediately. CPR is needed to keep oxygen supplied to the brain and other body organs until more advanced emergency medical treatment is available. Knowing how to immediately provide correct CPR procedures when confronted with an emergency can mean the difference between life and death for many victims. [Ref. 1]

Formal CPR classes are an important method for teaching these correct procedures Demonstrated public interest in learning CPR is high and more people are being trained in the correct procedures. However, in most cases when a cardiac arrest occurs, CPR is not started by bystanders, but is delayed until trained medical personnel arrive on the scene. This suggests that even people trained in correct CPR procedures may lack the confidence to use their skills in an emergency. [Ref. 2]

It is the intent of this thesis to provide an intelligent computer-assisted instruction system, not to replace currently available CPR courses, but to supplement them. This one-on-one tutoring is designed to raise the confidence level of the user so that when confronted with an emergency, critical minutes will not be lost. An intelligent computer-assisted instruction system can offer several unique advantages over the more traditional classroom approaches. The 24-hour availability of such a system affords the user the scheduling flexibility of receiving training at their convenience. The system,

unlike a human tutor or instructor, never loses patience with the student and can provide undivided attention at all times. The system also has the capability to test the student on unusual occurrences, many of which are more likely to occur when the student is allowed to pursue their own ideas rather than be streamlined through the correct or "logical" sequence of actions.

B. METHODOLOGY

In designing and implementing a computerized instruction system for CPR, henceforth referred to as the CPR Tutor, artificial intelligence (AI) techniques implemented in the Prolog programming language were utilized. A computer program such as this is referred to as intelligent computer-assisted instruction (ICAI) system or an intelligent tutoring system (ITS). Both terms will be used interchangeably throughout this study. The development of such a system focuses primarily on the problems of knowledge representation, student misconceptions, and inferencing [Ref. 3].

The basic components of the CPR Tutor are the student model, the expert module, the tutor module, and the user interface. The student model represents what the student does or does not know. The comparison of the student model with the expert module generates discrepancies that are analyzed by the tutor. The tutor provides appropriate feedback and updates the student model accordingly. The user interface serves as a buffer between the user and the CPR Tutor.

Programmed with information provided by the American Red Cross [Ref. 1], the CPR Tutor can randomly generate various emergency situations. The situation is displayed to the user as a list of facts. The CPR Tutor can prompt the user for an action in the stated situation, require a more detailed description of the action from the user, or provide assistance to the user in selecting the most correct action. After the user has entered their action selection in the given situation, the CPR Tutor will compare the

user's choice with that of the expert module. Based on this comparison, the user gets immediate results of their action in the form of a positive acknowledgement for the preferred response, a warning statement of varying severity for an incorrect response, or a message indicating the users response differs from the expert but the system will pursue the user's approach to the situation.

The CPR Tutor was organized into four primary files: *cpr*, *mecpr*, *metutor*, and *utilities*. The *cpr*, *mecpr*, and a portion of the *utilities* files were developed by the author to provide the CPR domain knowledge as well as the overall structure of the user interface. This Prolog code was developed to operate in conjunction with a generic means-ends analysis tutoring system, *metutor*, developed by Professor Rowe of the Naval Postgraduate School. Minor modifications were made to the *metutor* file to accommodate a menu-driven user interface and to include a limited amount of CPR domain specific tutorial responses in addition to the general tutoring strategies already present.

C. PREVIEW

Chapter II provides background information on CPR and the method by which it is currently taught by the American Red Cross.[1] Also included is a description of the the training format used in the American Red Cross Adult CPR course. Chapter III provides a background on computer-assisted instruction (CAI). This includes a review of computer-based instruction (CBI) and intelligent computer-assisted instruction (ICAI); these two are compared and contrasted, and specific applications of each are introduced. Chapter IV discusses the actual design and implementation of CPR Tutor. Chapter V discusses use of the CPR Tutor. Issues of expandability and portability of the system are addressed, as well as limitations and benefits of the system. Chapter VI is the summary

[1]See References 1 and 2 for a detailed description of these procedures.

and provides a final discussion of the research questions. Appendices A through C present user sessions. Demonstrations are offered at each of the three user levels: novice, intermediate, and advanced. Appendix D contains the Prolog source code for CPR Tutor developed by the author. Appendix E contains the Prolog source code developed by Professor Rowe.

II. OVERVIEW OF COMPUTER-ASSISTED INSTRUCTION

A. BACKGROUND

The use of the computer for teaching dates back to the late 1950s. The most common label for computer assistance in teaching and in the learning process is CAI, referring to computer-assisted instruction or, equally commonly, computer-aided instruction. CBI, computer-based instruction, is also used in reference to this same instruction. Other labels which are used include CAL, where "learning" replaces "instruction" and thus places a greater emphasis on the activities initiated by the learner than on the instructional materials created by the teacher-author. Replacing "instruction" with "education" as in CAE or CBE, computer-based education, similarly shifts the implication to infer a wider variety of computer uses, including administrative data processing and materials production as well as student use of computers. If emphasis is to be placed on the computer assisting the teacher in managing instruction, computer-managed instruction, CMI, is the more appropriate label. [Ref. 4]

Intelligent computer-assisted instruction, ICAI, refers to the more recent form of computer delivered instruction. ICAI utilizes artificial intelligence technology applied to instruction. To provide greater distinction and differentiation between this more recent approach and the older, more traditional ones, the descriptive phrases ICAI and CBI will be used throughout the remainder of this study.

B. COMPUTER-BASED INSTRUCTION

Traditional computer-based instructional systems tend to be designed and implemented by educational psychologists or technologists. Designed to solve practical problems by applying computer technology, a basic theme of CBI is that "the computer is

neither instruction nor a method of instruction; it is merely a vehicle of instruction". [Ref. 5] Although many different types of instructional strategies have been applied in CBI, the most prevalent have been based on the common approaches practiced in schools and training environments. The basic principle being that the instructor should successfully communicate their knowledge to the student to achieve the objectives within the imposed constraints. The success of the system is primarily determined by its degree of instructional effectiveness and efficiency. This teacher-centered approach requires the student to first understand the teacher's instruction and reinforce this understanding through practice. In this approach, the student has little or no initiative in the instructional process [Ref.3:pp.24-29].

The early CBI systems were developed primarily as a supplement to the principal instructional process. Taking the form of programmed-instruction paradigms, the formats of these systems were either electronic "page turners", which simply printed prepared text, or drill-and-practice, which printed problems and utilized prestored answers and remedial comments to respond to the student's solution [Ref.6:p.225]. As CBI has since evolved to become a main instructional delivery system, as opposed to merely a supplemental system, the format has diversified. In addition to the drill-and-practice approach, CBI formats now include tutorial, games, and simulations.

C. ICAI SYSTEMS

ICAI is a more recent form of computer-assisted instruction. Evolving from the field of computer science, it is an example of the application of artificial-intelligence technology to instruction. The basic philosophies underlying the structures and development processes are therefore fundamentally different from traditional CBI. The focus of ICAI projects has been on the technical and instructional aspects of the system as opposed to the domain features. Many ICAI projects have, however, also sought to

better understand the cognitive processes involved in learning and teaching. It can thus be asserted that ICAI programs actually lie at the intersection of computer science, cognitive psychology, and educational research. [Ref. 3]

1. Structural Methodology

Although ICAI programs often bear little outward resemblance to each other, most of these systems deal with similar issues and contain similar functional components. A typical ICAI system contains three primary components: the representation of knowledge, the provision of instruction, and the modeling of learning behavior. These components are commonly referred to as the expert module, the tutor module, and the student model respectively. In addition to these three components, a student interface is also necessary. Figure 2.1 depicts a generalization of the components in an ICAI system [Ref. 3].

a. Expert Module

The expert module is the encapsulation of the domain expert, containing the knowledge that the system is attempting to impart to the student. It is this module that generates "problems" and is used to evaluate the correctness of the student response. If a fully developed ICAI system is to be maximally effective, several experts in the domain should be members of the development team, to overcome the incomplete knowledge or conceptual vacuums that are often present in a sole expert. The domain knowledge found in textbooks is also often incomplete as well as idealized and, as such, is inappropriate as the primary knowledge source for a truly effective and operational ICAI system. [Ref. 7]

b. Tutor Module

A distinguishing feature of ICAI systems is the separation of the expert knowledge from the teaching strategies. An AI system that is an expert in a particular domain is not necessarily an expert teacher of the same domain. The acquisition of

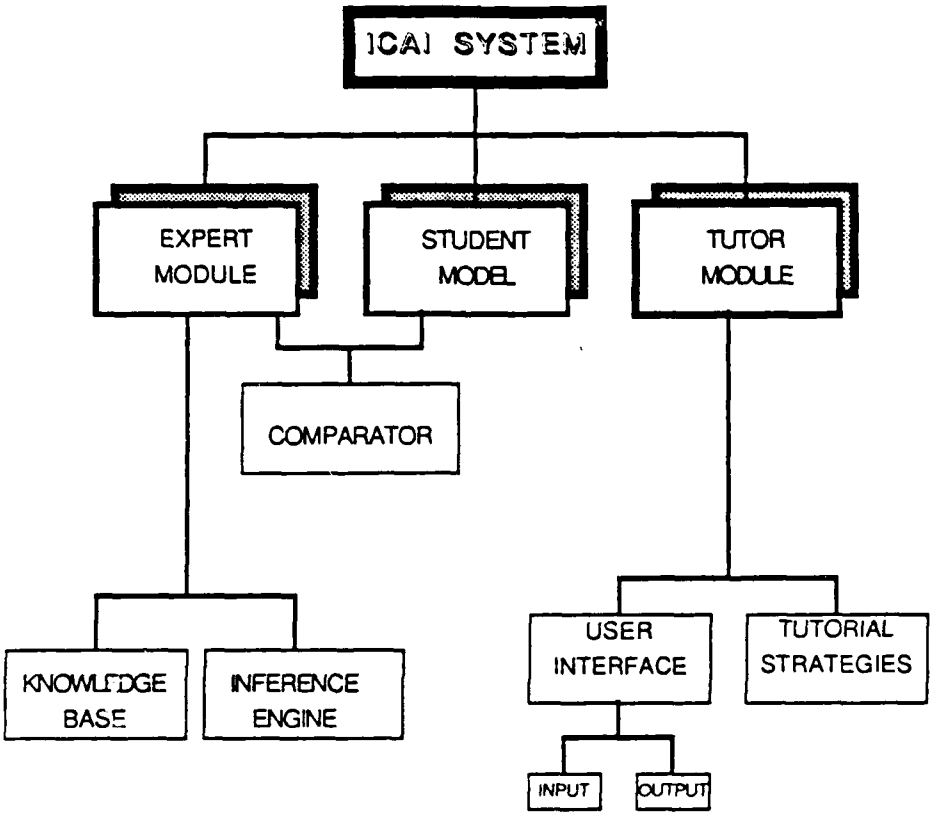


Figure 2.1 ICAI System Components

sufficient and correct teaching expertise has been a long-term problem for builders of tutoring systems [Ref. 7]. Multiple teaching strategies exist and no single strategy is appropriate for every domain. Each state in the domain, as well as each student, must be assessed if an appropriate teaching strategies are to be employed. It is therefore the task of the teaching expert, the tutor module, to choose the next strategy. This includes

providing diagnostics, questioning the student further or in more detail, and presenting new information. A central issue in the design of this module is the appropriateness of providing immediate responses to the student's action as opposed to temporarily allowing the student to continue down an incorrect path without interruption.

ICAI systems mainly use two types of instructional strategies although many other strategies do exist and have been used. The two basic classifications are the Socratic, tutorial method and the coaching method. The Socratic method involves questioning the student and guiding them through the process of debugging their own misconceptions. Wenger [Ref.8:p.39] summarizes this method by stating:

> In the Socratic method, the tutor does not teach a subject by direct exposition, but leads the student by successive questions to formulate general principle on the basis of individual cases, to examine the validity of her own hypothesis, to discover contradictions, and finally to extract correct inferences from the facts she knows.

The student is thus encouraged to reason about what they know and thereby modify their conceptions.

The goal of coaching is to encourage the acquisition of skills and general problem-solving abilities by engaging the student in activities such as computer games. The immediate aim of the student is to have fun; skill acquisition and learning is, by design, an indirect consequence. Tutoring comes about through appropriate interruptions by the coach that offer new information or suggest new strategies. Primary challenges in using this instructional method lie in being able to "observe" the student's play of the game, determining what skills or knowledge the student is likely to acquire based on their playing skills, and judging effective times and means for intercession. [Ref.6:pp.233-235]

c. Student Model

The primary focus of the student model is to represent what the student does and does not know, or the student's understanding of the material being taught. The

purpose of modeling the student is to enable specific assessments to be made of the student's knowledge of the domain and to hypothesize about their misconceptions and the reasoning strategies they employed. The tutor module is thereby able to point out misconceptions and make suggestions. The model is generally constructed by comparing the student's response in a given state to that of the expert in the same state, the expert being contained in the expert module. This techniques is referred to as the "overlay model." [Ref.3:p.17]

2. Applications

The impetus of the development of ICAI systems was **SCHOLAR** , a Socratic tutor designed to teach students about South American geography. It was the pioneering effort in the development of tutorial dialogues capable of handling unanticipated student questions and of generating instructional materials in varying levels for the student. Developed in the 1970, SCHOLAR is on of the earliest ICAI systems and, as such, was instrumental in setting the basic themes and goals for the field. [Refs.6, 8]

An extension of SCHOLAR, **WHY** tutors students in the causes of rainfall. Also implementing the Socratic tutoring method, WHY made the effort to describe the global strategies used by human tutors to guide the tutorial dialogue. Although it remained primarily theoretical, WHY represented the first attempt at capturing a complete tutorial strategy as complex as the Socratic method in a concise, rule-based computational model. [Refs.6, 8]

SOPHIE was developed to explore the student initiative during the tutorial interaction. The pedagogical purpose in SOPHIE was to provide a reactive learning environment in which the student could try out their own ideas, have them critiqued, and receive advice. Set in the domain of troubleshooting electronic circuits, this program allowed the student the chance to apply their theoretical knowledge of electronic laws in

a simulated electronics laboratory. Students were challenged to explore their own ideas and to come up with conjectures or hypotheses about troubleshooting strategies for electronic circuits. The role of the expert was to provide detailed feedback as to the logical validity of their proposed solutions. With roots going back to the early 1970s, SOPHIE went through three successive phases spanning more than five years. The research it spawned continues today. [Refs.8, 9]

Initiated in the context of SOPHIE, which was originally meant to include a troubleshooting coach, **WEST** was the first ICAI system to use the coaching strategy. The domain of electronics was viewed as too complex for a first computer-based investigation into the strategy of coaching, hence WEST was developed to coach students in elementary arithmetic. Another ICAI system that used the coaching strategy was **WUMPUS**, which challenged the student in the domains of logic, probability, decision theory and geometry. [Refs.6, 9]

GUIDON was a program developed to teach diagnostic problem-solving. Its mixed-initiative dialogue differs from the other ICAI systems listed above in its use of prolonged, structural teaching. It went beyond simply responding to the student's last move, the methodology used in WEST and WUMPUS, and the repetitive questioning and answering technique used in SCHOLAR and WHY. The uniqueness of GUIDON, however, was in its attempt to turn an existing expert system, MYCIN, into an intelligent tutor. GUIDON demonstrated the inappropriateness of using just an expert system's knowledge base as the expert module of an ICAI system without supplementation to help explain and organize the knowledge in the process of teaching. [Refs.6, 9]

While a complete ICAI system has the three primary components, not all systems do. **PROUST** and **STEAMER** are two such examples: they do not contain tutoring functions. PROUST is a knowledge-based system to find nonsyntactic bugs in

Pascal programs written by beginning programmers. The key idea of STEAMER is the conception of an interactive inspectable simulation of a steam propulsion system. Its primary concern is how people understand and reason about complex systems as well as how interactive graphical interfaces might support the development of useful mental models. Started in the late 1970s, STEAMER is not yet an operational system. However, it is considered of significant importance in the field of ICAI because it called for a very careful look at what aspects of AI technology are actually ready for application in the design and implementation of an instructional system. [Ref. 3]

OBIE-1:KNOBE was the first generic system to enable the construction of realistic, dynamic representations of real devices. Intelligent simulations previous to it were not generalized. They were specifically designed and built with only the immediate subject domain in mind. [Ref. 10]

One system which has been fully implemented, tested and is in use, although still being augmented based on formal student evaluations, in nearly sixty industrial sites across the United States is **RBT**, Recovery Boiler Tutor. The tutor is based on a mathematical formulation of a type of boiler found in paper mills. It provides an interactive simulation in which emergencies can result from inappropriate student responses. An unusual point about this system is its use of Fortran and C as the implementation languages. [Ref. 7]

D. COMPARISON AND CONTRAST OF CBI AND ICAI

1. Focus

Traditional CBI has been developed primarily by educational psychologists and technologists. The role of the computer has generally been that of a vehicle for efficient implementation. As an older, more established field, textbooks are available which suggest detailed guidelines, procedures and principles which should be followed in

designing a CBI system. The overall goal of a CBI system is thus the usability of an effective and efficient system [Ref. 5].

ICAI, in contrast, has primarily been a computer science initiative in the application of AI technology in the educational/instructional process. The focus has been on the technical aspects such as knowledge representation techniques, inferencing mechanisms, natural language dialogues, etc.

2. Subject Matter Domain

CBI systems have been developed for a wide range of subject areas, including math and sciences as well as the arts. The subject matter is, in fact, virtually unlimited. The application domain of ICAI systems have generally been limited to subject areas which are themselves relatively well-structured. With the emphasis being on the AI technology rather than a specific domain, subject matter areas have primarily been selected due to their appropriateness for the AI technique being researched. The emphasis has thus been more on the process itself, rather than the end result. [Ref.3:p.28]

3. Format

The early CBI programs were used generally as a supplement to the principal instructional method. The basic format was drill and practice, serving to reinforce the material that was previously presented by the instructor. CBI has since evolved into a main instructional delivery system and the format has correspondingly diversified. Common CBI formats now include intrinsic and extrinsic games; physical, situational and process simulations; and tutorials in addition to drill and practice. [Ref.3:p.27]

ICAI does not tend to use the simulation nor the drill and practice formats. Instead, these systems tend to fall into the category of either a tutorial or a game. Although both CBI and ICAI systems use these latter two formats, the methodology and

purpose are different. In a CBI tutorial, the questions are used to reinforce the material that has been provided by computer-based instruction. These questions are always initiated by the system, answered by the student, and are directly related to the presented material. The computer does not "know" the subject matter but merely presents blocks of text. Questions are posed to the student based on a predetermined algorithmic process. ICAI systems, on the other hand, take a more dynamic approach. Attempting to carry on a series of question and answer exchanges, many ICAI systems can carry on a natural language, or pseudo-natural language, dialogue with the student. Not only does the system ask questions of the student, but the student is also able to direct questions to the system. In addition to allowing greater flexibility, ICAI systems utilize these question-and-answer exchanges to make a determination of what the student understands and direct the instructional process accordingly. [Ref.3:pp.24-27]

Intrinsic games are used in CBI as a means of teaching gaming rules and/or skills. Extrinsic games are used primarily to maintain the motivation of the student. The use of games in ICAI systems is similar to the use of extrinsic games in CBI systems, the purpose being to provide a "reactive learning environment". The difference is that in ICAI systems, students are expected to develop a higher level of knowledge. The games are generally designed to allow the student the ability to explore and test their own ideas and thus, control the instructional process to a greater degree. [Ref.3:pp.27-28]

4. System Design Structure

Traditional CBI structure is frame-oriented. It is organized as a file of screens to be presented to the student. A more dynamic approach is taken by an ICAI system. The design structure is the combination of a knowledge database with a transaction series for using the database [Ref. 11]. Thus, a distinguishing feature of ICAI systems is the separation of the teaching strategy(s) from the subject expertise to be taught [Ref. 6].

5. Development Process

The development of most CBI programs has utilized a systems approach [Ref. 5]. This approach requires several steps of procedural activities. Included in this process is a needs analysis, followed by the design and development as well as the implementation. Evaluations are conducted throughout the various stages.

No such generalized procedure can describe the development of ICAI systems. In addition to the fact that very few ICAI systems have been chronologically or systematically documented, the inherent difference in goals plays a significant role. The development of each ICAI system has been guided by the specific goals and skills of the individual researchers as well as the actual characteristics of the application domain. The development process has, therefore, varied among the various ICAI projects.

In comparing the development time required for CBI and ICAI, it is estimated that the first hour of a CBI system requires approximately two hundred hours of programmer preparation, an amount probably exceeded by the preparation time required for an ICAI system. Each additional hour of instruction in a CBI system requires an additional two hundred hours of programmer preparation. ICAI systems, by comparison, are designed to utilize a single piece of knowledge in many ways. Therefore, each additional hour of instruction can be expected to require relatively less programmer preparation time. [Ref. 7]

6. Hardware and Software

A driving force in the development of computer delivered instruction has always been the evolution of computer technology. CBI systems of the 1960s and 1970s utilized special computer systems developed primarily for CBI, such as PLATO, TICCIT, and IBM 1500. Each of these early systems also has their own programming language specifically designed for authoring CBI systems. These were, respectively,

TUTOR, APT/TAL and Coursewriter. These systems, with the exception of IBM 1500, are still in use today. With much lower associated costs, microcomputers, however, have become the most commonly used system for both development and implementation. In addition to the system-specific languages, three other levels of software are currently used. These are: 1) general-purpose languages such a Pascal, C, FORTRAN and BASIC, 2) system-independent CBI languages, and 3) authoring systems that provide facilities, without requiring programming skills to develop CBI courses of instruction. Although exceptions exist, ICAI systems, with their emphasis on the application of AI technology, have primarily been developed on special AI hardware or computers supporting an AI language. The software has been, for the most part, limited to LISP and Prolog. [Ref.3:pp.29-30]

III. OVERVIEW OF CPR

It is estimated that one out of every five Americans has some form of cardiovascular disease and one out of every two people will die from a heart attack or a related disease. As such, it is the leading cause of death in the United States. It is estimated that forty percent of cardiac arrest victims could be saved if CPR was immediately provided and the initiation of advanced life support measures by qualified medical personnel followed within eight to ten minutes. [Ref. 1]

A. PURPOSE

Cardiopulmonary resuscitation is the combination of artificial respiration and manual artificial circulation. It is a means of supplying oxygen to the body when a person's heart has stopped. By breathing oxygen into the victim's lungs, oxidated blood can then be circulated through the victim's body by manually compressing and releasing on the lower half of the sternum. CPR does not "revive" victims nor does it restart the heart. CPR does, however, provide critical life-sustaining oxygen to the brain and other body organs until advanced life support systems provided by professional medical personnel can get the heart restarted. Thus, the twofold purpose of CPR is 1) keeping the lungs supplied with oxygen when breathing has stopped and 2) keeping blood circulating and carrying oxygen to the brain, heart, and other body parts. [Ref. 1]

B. PROCEDURES

Cardiopulmonary resuscitation is a combination of rescue breathing in which oxygen is artificially breathed into the victim and manual chest compressions which keep oxygen-carrying blood flowing through the blood vessels. The external cardiac compressions consist of the application of rhythmic pressure over the lower half of the

sternum, compressing the heart and producing a pulsatile artificial circulation. Artificial ventilation achieved through rescue breathing must always be provided in conjunction with external cardiac compressions. [Ref. 1]

To achieve maximum effectiveness in the performance of CPR, actions by the rescuer must be deliberate and precise. Attention to detail is vital. The order of actions and the number of repetitions is not arbitrary. The following paragraphs list the actions, henceforth referred to as operators, which are the basics of CPR. Figures 3.1 and 3.2 summarize the sequencing of actions.

Check unconsciousness is the first operator recommended in the process of providing CPR. The determination of consciousness or unconsciousness will dictate which further actions may be required. Although a conscious person may require first aid, CPR is not applicable. If the person should lose consciousness and no pulse nor breathing is observed, CPR should be initiated immediately.

Yell for help is performed to beckon assistance. If assistance is available, they can be instructed to phone emergency medical services. If assistance is not available, it will be necessary for the rescuer to perform CPR for at least one minute and then temporarily cease CPR to contact the paramedics personally. This operator is ideally performed after unconsciousness is checked. However, it can also be correctly applied prior to the check unconsciousness operator.

Position victim is a prerequisite to to the proper performance of CPR. It is impossible to accurately judge whether or not a victim is breathing unless they are on their back. Chest compressions also require the victim to be on their back.

For breaths provided by the resuer to be effective, the airway must be open. Therefore, it is necessary for the rescuer to perform the operator **open airway.**

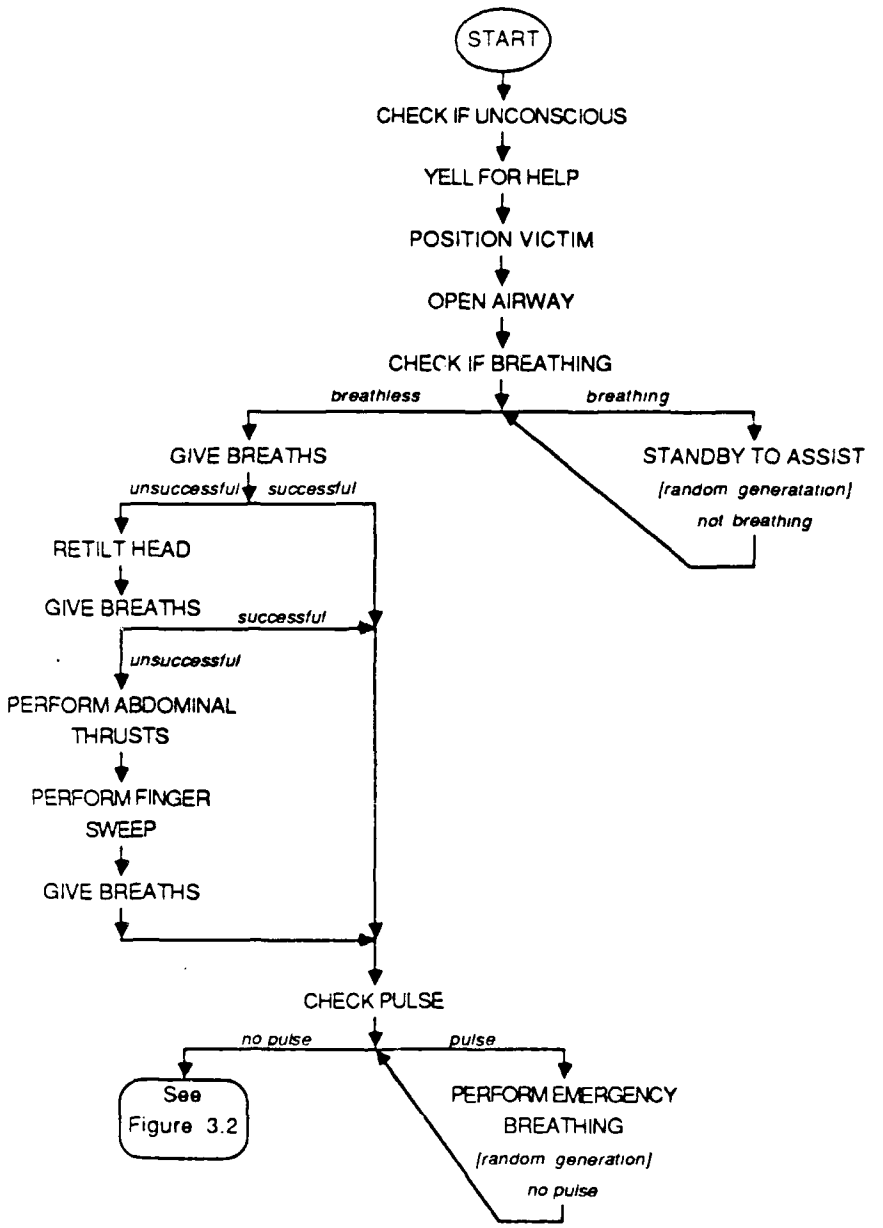


Figure 3.1 CPR Procedural Flowchart - Part One

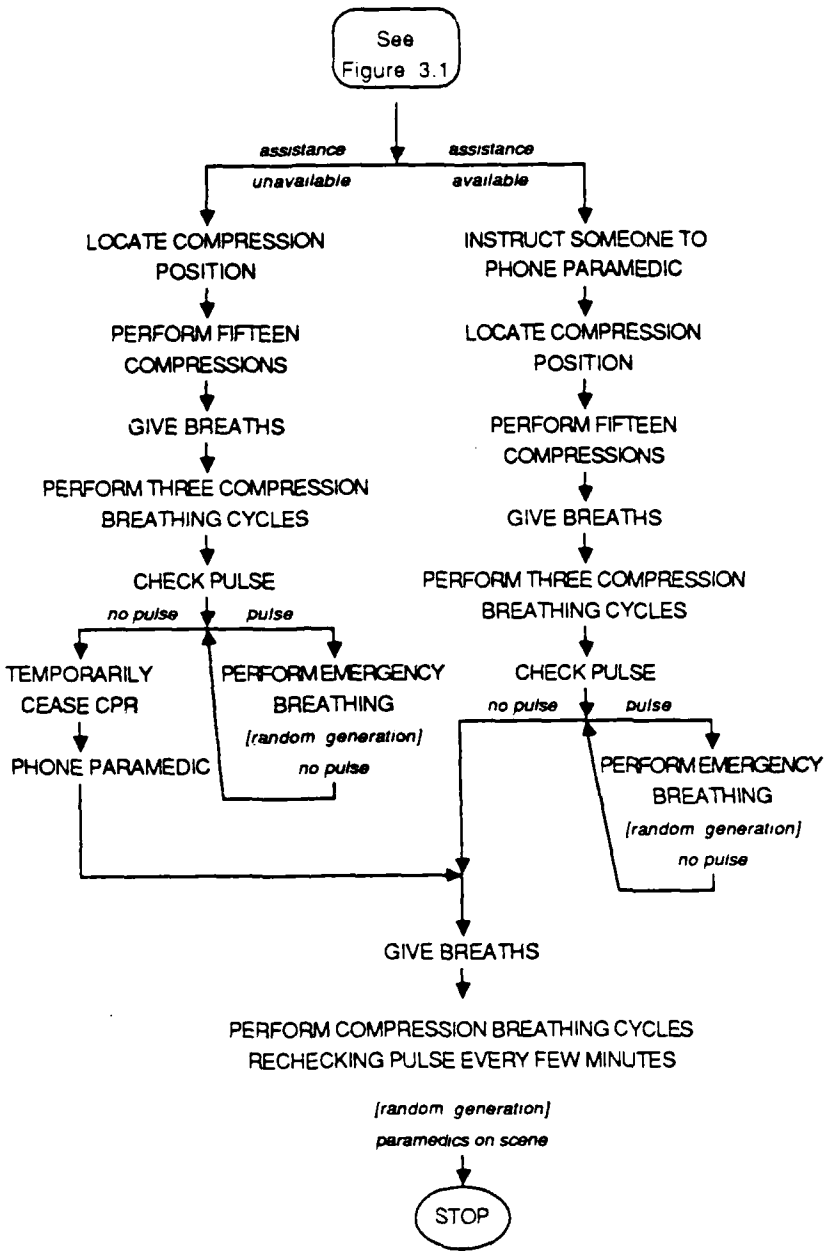


Figure 3.2 CPR Procedural Flowchart - Part Two

If a person is breathing, CPR is never applicable. In fact, performing chest compressions on a breathing victim may result in serious injuries. Prior to performing CPR on an unconscious person it is therefore necessary to **check for breathing.**

Standby to assist is not an actual CPR operator. It is performed only as a default. It is used when the victim is breathing and is not bleeding.

The operator **give two breaths** is performed multiple times throughout the CPR process. When breathing ceases, artificial breathing by the rescuer is necessary to maintain a flow of oxygen within the victim's body. Oxygen is forced through the body by compressions performed by the resuer on the sternum. To be effective, the victim's airway must be open and unobstructed.

If breaths into the victim do not successfully go in, it is necessary for the rescuer to use the operator **retilt head.** This is the first action to be taken in attempt to successfully enable oxygen to enter the victim's body. It is followed by a second attempt of the give two breaths operator.

In the event of an obstructed airway, **perform abdominal thrusts** is an operator to dislodge a foreign object. Abdominal thrusts are performed after first retilting the victim's head and a second attempt to provide two breaths is unsuccessful. After performing abdominal thrusts, the object, if present, should be removed by performing a finger sweep.

Perform finger sweep is conducted following the abdominal thrusts. The intent is to dislodge a possible object which may be blocking the victim's airway and preventing oxygen from entering the body.

The operator **check pulse** is a necessary step in the CPR procedure because performing chest compressions on a victim who has a pulse can result in serious medical complications. Emergency breathing should be provided to a victim who has a pulse but

is not breathing. For the victim who is not breathing and who does not have a pulse, CPR procedures should be initiated immediately.

When the heart ceases beating, blood no longer circulates, which deprives the body of oxygen. The victim will stop breathing due to a shutdown of the respiratory system. A person with no pulse will thus not be breathing. However, a person who is not breathing may have a pulse. In this case, it is necessary for the rescuer to **perform emergency breathing.** This is a high level operator which actually consists of the repetitive use of give two breaths operator as well as the recheck pulse operator. This cycle is performed until such time as the victim is breathing on their own or their pulse has also stopped, in which case chest compressions associated with CPR become applicable.

One of the two options for contacting professional medical services is **instruct someone to phone paramedic.** If assistance is available, they should be instructed to contact professional emergency medical services after the resuer has determined the vital signs of the victim.

The second option for contacting professional medical services is for the rescuer themselves to **phone paramedic.** If assistance is not available, it is necessary for the rescuer to first perform CPR for at least one minute and then temporarily cease CPR to contact professional emergency medical services. CPR should then be immediately resumed.

Temporarily cease CPR is not a CPR operator per se. It is performed by the rescuer if assistance is not available and they must personally contact professional medical services. As such, it is only performed in conjunction with the phone paramedic operator.

Locate compression position is a prerequisite to the performance of chest compressions. Not locating the correct position could nullify the significance of performing chest compressions.

Perform fifteen compressions causes the the oxygen the rescuer has breathed into the victim to be forced throughout the body. Prior to performing these compressions, the correct compression position must be located.

The essence of CPR is **perform compression/breathing cycles.** This consists of the rescuer providing breaths to the victim, followed by chest compressions to force the oxygen through the victim's body. Chest compressions should never be performed on a person with a pulse. Therefore, periodic checks for a pulse should be performed. This higher level operator is used in the latter portion of the CPR procedure. It's used in lieu of multiple instantiations of the two low level operators, perform fifteen compressions and give two breaths.

Recheck pulse is performed to ensure chest compressions are not performed on a victim with a pulse. If compressions are performed on a person with a pulse, serious medical complications may arise.

Stop bleeding is not actually a CPR operator, but is included to provide more realism. It is important for a rescuer to realize that CPR always takes precedence over other first aid actions. It is performed only if the victim is bleeding and is also breathing. The rescuer must be primarily concerned with the pulse and breathing of the victim. Bleeding must be ignored if the victim has no pulse rate and/or is not breathing.

C. TRAINING PROGRAMS

The American Red Cross Adult CPR course was designed for the general public. Anyone thirteen years of age or older or who has completed the seventh grade and expresses an interest in learning CPR procedures is encouraged to take the course. The

duration of the course is six hours and can be taught in either one session or split into two three hour sessions. To maintain Red Cross certification, retraining is required on an annual basis. The course includes three components: a student workbook, six films, and practice sessions. Participants read the workbook and answer review questions at the end of each chapter. Films are interspersed throughout the course. They include real-life emergencies and show demonstrations of the first aid skills being taught. Practice sessions and skill tests allow students to actually practice the learned skills. Each student must also pass a written test to receive a course completion certificate. [Ref. 1]

IV. CPR TRAINING: AN ICAI APPROACH

A. SYSTEMS ORGANIZATION

The CPR Tutor is an ICAI system which utilizes the Socratic tutoring methodology. It was developed and implemented on a Digital Equipment Corporation VAX minicomputer using the C-Prolog language. The design is based on the interaction of the four primary modules discussed in chapter two, section C. These are:

1. Expert Module
2. Tutor Module
3. Student Model
4. User Interface

Figure 4.1 illustrates the interrelationships of these modules. Also shown is their file location in the CPR Tutor program.[2]

B. EXPERT MODULE

The expert module of an ICAI system consists of two major parts. These are the inference engine and the knowledge base.

1. Inference Engine

The inference engine used in the CPR Tutor prototype is an application of **means-ends analysis**, a classic technique used to solve search problems by abstraction. Using double recursion to decompose the CPR procedure, means-ends analysis reasons top-down from abstract goals, with the "ends justifying the means." The means-ends analysis Prolog program used in the CPR Tutor prototype was written by Professor

---

[2]See Appendices D and E for a listing of these files.

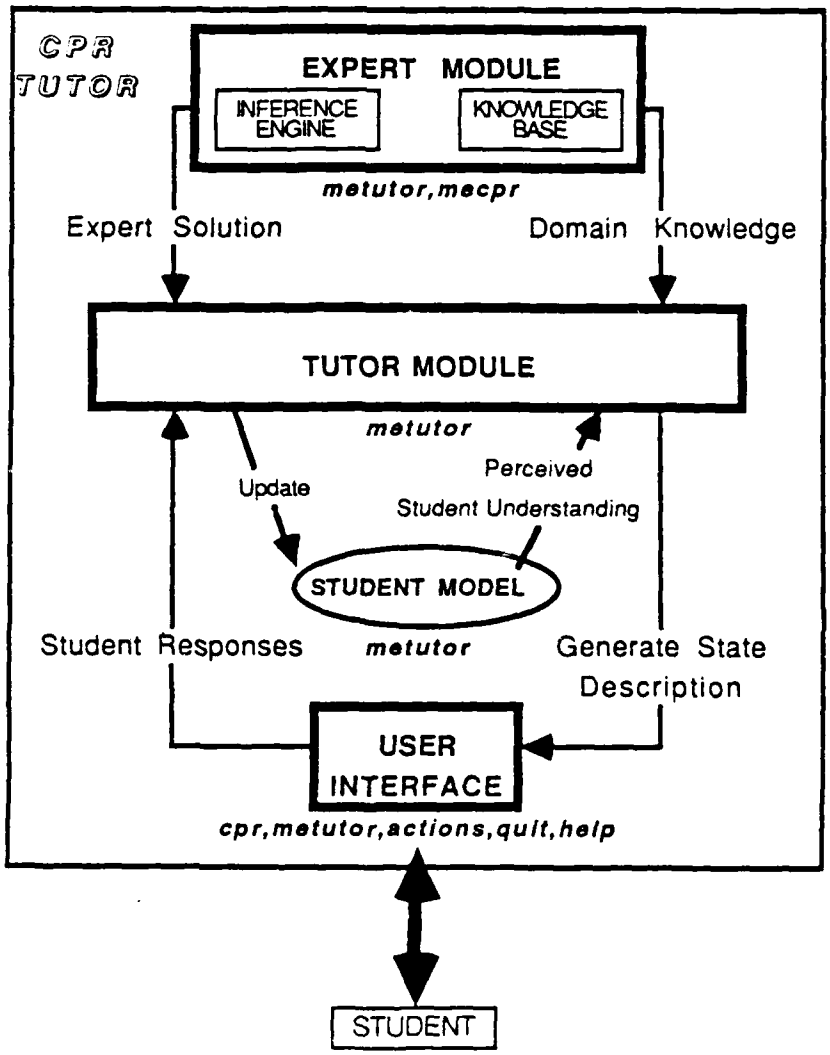


Figure 4.1 CPR Tutor Structural Model

Rowe. The top-level rule is a recursive **means-ends** predicate of four arguments: **State, Goal, Oplist** and **Goalstate.** The **State** is a complete list of the facts which are true in a state. In the CPR Tutor prototype the single fact *person-collapsed* was chosen to be the starting state. The **Goal** is a list of facts that are required to be true in the goal state (or partial description of the goal state). The **Goal** chosen for the CPR application was *victim is breathing and_or paramedic is on scene*. The **Oplist** is the list of operators that are required to reach the goal state from the starting state. This list is always initially the same. However, it changes as random substitutions of facts are made. The **Goalstate** is the complete list of facts which are true in the goal state. [Ref. 12]

Professor Rowe [Ref.12:p.270] describes the operation of the means-ends program as follows:

> This recursive program has a single basis step (the first line), which says we don't need any operators to solve a problem in which the starting state includes all the goal facts. The rest of the lines are a single induction step, which...has two recursive calls: the first for the preconditions, the second for the postconditions. The lines say to first compute the list of facts different between **State** and **Goal**, and find a recommended operator for some subset of those facts. Look up the preconditions of the operator and recursively call **means-ends** to figure how to satisfy them. Then look up the deletion postconditions and delete them from the final state resulting from the precondition recursion; look up the addition postconditions, and add them to that state. Now we have the state after application of the original recommended operator. So recursively call **means-ends** to figure how to get from here to the original goal. The final operator list for the whole problem is the appending together of the precondition-recursion operator list.

A complete listing of this unchanged program used as the inference engine in the expert module in CPR Tutor is contained in Appendix E.

2. Knowledge Base

Cardiopulmonary resuscitation is a combination of artificial respiration and manual artificial circulation. These two primary actions can be further decomposed into a series of basic sub-actions, the application of which is not arbitrarily ordered. CPR can be considered a procedural evolution from a starting state to a goal state via intermediary states. Because most of these abstracted states are strongly ordered in the CPR process,

there exists nearly a unique recommended solution path, i.e., recommended application of operators from the start state to the goal state. Deviations from this path in the means-ends program can occur only as a result of a random side effect. One example of such a random change in state is a deletion of the fact *victim is breathing* and the addition of the fact *victim is not breathing*. When the victim is breathing no action, other than to standby to assist, is required of the rescuer. However, it is necessary that the rescuer be knowledgeable of what actions are necessary should the victim cease breathing. The random generation indicated above is designed to test the student's knowledge of what to do in this event and prompts the continuance of the tutoring session.

The knowledge-base operators must be clearly identifiable and there must exist a recommended ordering among them. This recommended order, however, need not be predetermined by a list. Instead, each operator has one or more associated **recommended** predicates:

recommended([<resultant-fact>], <operator>).

which indicate the facts that application of the operator will help achieve.

Also associated with each operator must be a **precondition** fact and a **postcondition** fact.[3] Preconditions are the prerequisites required for the application of the operator. Postconditions are the changes that occur when an operator is applied.

Two-argument facts identify preconditions. The first argument is the operator and the second argument is the list of applicable prerequisites for the application of the

[3]The recommended, precondition, and postcondition predicates are necessary input to Professor Rowe's means-ends analysis program which serves as the expert module's inference engine. As such, the formats of each are predetermined.

specified operator. The format for precondition facts is as follows:

precondition(<operator>,[<prereq-fact1>,<prereq-fact2>,....]).

If no precondition exists for an operator, i.e., it can be applied at any time, the null condition, [], is used. The precondition facts used in the CPR Tutor prototype are displayed in Figure 4.2.[4]

Postconditions can either add new facts, delete facts, do both, or do neither. Two kinds of postconditions are thus associated with each operator, addpostconditions

---

```
precondition(check_if_breathing,
    [on_back(victim), open(airway), not(breathing(victim)), not(stopped_breathing)]).
precondition(recheck_pulse,
    [unconscious(victim), performed('compression breathing cycles')]).
precondition(open_airway, [on_back(victim), unconscious(victim)]).
precondition(perform_abdominal_thrusts, [retilt(head), given('first pair of breaths')]).
precondition(perform_finger_sweep, [performed('abdominal thrusts')]).
precondition(perform_emergency_breathing,
    [unconscious(victim), observed('victim not breathing'), observed(pulse)]).
precondition(perform_three_compression_breathing_cycles,
    [performed('fifteen compressions'), given('second pair of breaths')]).
precondition(phone_paramedic, [unavailable(assistance), temporarily_stopped(cpr)]).
precondition(stop_bleeding, [bleeding(victim)]).
precondition(yell_for_help, []).
```

Figure 4.2 CPR Tutor Sample Preconditions

---

[4]See also Appendix D.

and deletepostconditions. The basic format of each is similar to that for preconditions:

addpostcondition(<operator>,[<assert-fact1>,<assert-fact2>,...]).

deletepostcondition(<operator>,[<delete-fact1>,<delete-fact2>,...]).

Real world applications, however, are not always so cut-and-dried. In many applications, including CPR, the same operator can have different effects in different circumstances. An example is the application of the operator *give breaths*. This operator is applied numerous times in the the process of providing CPR. It is necessary to assert which iteration of *give breaths* has been applied each time the operator is employed. To allow this flexibility, multiple three-argument postcondition predicates can be used for applicable operators, in addition to the standard two argument format.[5] The format of this three-argument form is:

addpostcondition(<operator>,[<prereq-fact1ist>],[<assert-factlist>]).

deletepostcondition(<operator>,[<prereq-fact1ist>],[<delete-factlist>]).

Figure 4.3 contains a sampling of the various two and three-argument postcondition facts used in the CPR Tutor.

To more accurately reflect real-life circumstances it necessary to allow the possibility of randomness. For example, when the rescuer checks the victim's pulse, a pulse may or may not be present. Thus, although each operator has obligatory postcondition facts, other facts can be deleted and/or inserted in the new state through the

[5]This enhancement was developed by Professor Rowe to allow increased flexibility.

```
deletepostcondition(open_airway, []).
addpostcondition(open_airway, [open(airway)]).

deletepostcondition(perform_emergency_breathing, [observed(pulse)]).
addpostcondition(perform_emergency_breathing,
   [given('first pair of breaths')], [observed('no pulse')]).
addpostcondition(perform_emergency_breathing,
   [not(given('first pair of breaths'))], [still_no_pulse]).
addpostcondition(perform_emergency_breathing,
   [error-in-perform-emergency-breathing]).

deletepostcondition(perform_three_compression_breathing_cycles,
   [performed('fifteen compressions'), given('second pair of breaths')]).
addpostcondition(perform_three_compression_breathing_cycles,
   [performed('compression breathing cycles')]).

deletepostcondition(give_two_breaths,
  [stopped_breathing, performed('finger sweep')])
addpostcondition(give_two_breaths,
  [phoned(paramedic), still_no_pulse],
  [given('sixth pair of breaths')]).
addpostcondition(give_two_breaths,
  [on_back(victim), open(airway), performed('fifteen compressions')],
  [given('second pair of breaths')]).
addpostcondition(give_two_breaths,
  [open(airway), on_back(victim), not(phoned(paramedic))],
  [given('first pair of breaths')]).
addpostcondition(give_two_breaths, [error-in-give-two-breaths]).

deletepostcondition(recheck_pulse, [phoned(paramedic)],
    [performed('compression breathing cycles')]).
deletepostcondition(recheck_pulse, []).
addpostcondition(recheck_pulse, [still_no_pulse]).
```

Figure 4.3 CPR Tutor Sample Postconditions

predicate **randsubst.**[6] The format for this predicate is as follows:

randsubst(<operator>,[[<delete-fact1>,<assert-fact1>,<probability1>,<optional-msg1>],

[delete-fact2>,<assert-fact2>,<probability2>,<optional-msg2>],...]).

Figure 4.4 contains a listing of the **randsubst** facts used in the CPR Tutor.

```
randsubst(check_pulse, [[observed('no pulse'), observed(pulse), 0.2]]).
randsubst(open_airway, [[unavailable(assistance), available(assistance), 0.3,
    'Someone has responded to your yell for help.']]).
randsubst(perform_emergency_breathing,
    [[unconscious(victim), unconscious(victim), 1.0,
    'The heart has stopped beating. What should you do now?']]).
randsubst(position_victim, [[unavailable(assistance), available(assistance),
    0.3, 'Someone has responded to your yell for help.']]).
randsubst(standby_to_assist, [[observed('victim not breathing'),
    observed('victim not breathing'), 1.0,
    'Breathing has stopped. What should you do now?']]).
randsubst(stop_bleeding, [[breathing(victim), observed('victim not breathing'),
    0.4, 'Bleeding is stopped. However, the victim has stopped breathing.']]).
randsubst(yell_for_help,[[unavailable(assistance), available(assistance), 0.3,
    'Someone has responded to your yell for help.']]).
randsubst(give_two_breaths,
    [[given('first pair of breaths'), obstructed(airway), 0.3,
    'Airway is obstructed - you are unable to breathe air into the victim.']]).
```

Figure 4.4 CPR Tutor Random Substitutions

[6]This was an enhancement developed by Professor Rowe to allow for randomness.

C. KNOWLEDGE ACQUISITION

The CPR domain expertise was derived solely from a subset of the workbook material utilized by the American Red Cross to conduct formal CPR classes. Although this represents a reliance on a single expert instead of multiple experts as recommended, it was deemed adequate for the development of the CPR Tutor prototype. This decision was based on the prototypical nature of the CPR Tutor, the relatively limited length of available time for its development, and the fact that many medical experts were consulted in the formulation of the workbook itself. A operational version of the CPR Tutor, however, would benefit from the direct expertise of the actual medical experts.

Within the workbook, a skill test is provided which tests the student on their correct application of the CPR operators to a predefined scenario. This scenario unfolds as the student proceeds. For example, when the student performs the operator *check for breathlessness*, the instructor says "No breathing". The students acknowledges by repeating the phrase "No breathing".

Randomness, however, is not a factor in the American Red Cross skill test. The instructor does not, for example, arbitrarily select "No breathing" as opposed to "Breathing" as the response to give to the student. The direction is clearly defined by a script type format in which the instructor's response is given. For the development of the CPR Tutor, however, these junctures provided the most logical points for randomness. The expertise on how to respond to a direction other than the one taken in the skill test was derived from reading material provided elsewhere in the workbook and/or in the instructor's manual. For example, the skill test follows the assumption that someone else is available to assist the primary rescuer, i.e., the user of the CPR Tutor. Therefore, *instruct someone to phone paramedic* is one of the predefined operators that the student needs to perform to successfully pass the skill test. In textual material which follows the

skill test, the student is provided the information on what to do if no one responds the the rescuer's yell for help. It does not merely list operators which should be performed, but instead states the rescuer should perform CPR, i.e., compression-breathing cycles, for at least one minute prior to going to a phone themselves to call the paramedic. The rescuer should resume CPR as soon as possible thereafter [Ref. 1].

In transposing this expert knowledge into the means-ends analysis format required by the CPR Tutor, it was necessary to first define the applicable operators and their corresponding recommended, precondition, and postcondition facts. The process of creating these facts can perhaps best be explained by way of a couple examples.

One of the more straightforward operators was *open airway*. To open the victim's airway, it is physically necessary for the victim to be on their back. It is also not logical to perform this operator on anyone who is not unconscious. The precondition:

precondition(open_airway, [on_back(victim), unconscious(victim)]).

was used to capture these facts. Opening the airway does not delete any previously asserted facts, but its performance does presumably cause an open airway. Thus the following recommended and postcondition facts are applicable:

recommended([open(airway)],open_airway).

deletepostcondition(open_airway, []).

addpostcondition(open_airway, [open(airway)]).

Because this is one of the earliest operators performed, it is possible that someone could respond to the rescuer's earlier yell for help while this operator is being performed. With an assigned thirty percent probability of someone responding to the earlier yell for help

during the performance of the open airway operator, the following random substitution fact was created:

randsubst(open_airway, [[unavailable(assistance), available(assistance), 0.3, 'Someone has responded to your yell for help.']]).

When randomness was employed, complications in defining the means-ends analysis facts arose. For example, if assistance is available, this other person should be instructed to phone the paramedic after the victim's pulse has been checked and prior to the rescuer locating the compression position. This implied that a fact such as *help is phoned* should be a precondition for the compression position being located. However, this could not be the case because, if help did not respond, the *help is phoned* fact would not be applicable. In fact, it would be a wrong action on the student's part. Therefore, the precondition:

precondition(locate_compression_position, [observed('no pulse'), observed('victim not breathing'), not(available(assistance))]).

was used. The fact *not(available(assistance))* was used to represent the fact that help, if previously available, was no longer available at that point because the available assistant had left to phone the paramedic. This was a result of the deletepostcondition fact associated with the operator *instruct someone to phone paramedic*. If help had never responded to the *yell for help* operator, the precondition would also be met.

Thus, in representing the knowledge contained in the American Red Cross CPR workbook [Ref. 1], similar assignments were made for each of the operators. Some proved to be very straightforward in their translation into means-ends analysis facts.

Others, however, required adjustments in the code from what originally might have seemed sufficient. This was directly attributable to the implementation of randomness.

D. TUTOR MODULE

The tutor module[7] represents a distinguishing feature of ICAI systems over the more traditional CBI approach, in that the teaching strategies are separated from the expert knowledge base. In the CPR Tutor prototype, the tutor module is basically a stand-alone system which could be utilized to tutor any domain which can be represented as a search problem for means-ends analysis. For example, a more general version of this same tutor has been used to provide training to a Navy fire team leader in combatting a fire aboard a Navy vessel.

The purpose of the tutor module is to monitor the status of the student model and thus guide the course of future dialogue between the ICAI system for CPR and the user. To meet this purpose, it is necessary for the inclusion of information about effective teaching strategies in the tutor module. The tutoring rules used in this module are handled by the five argument predicate **handle_student_op**. The major tutoring strategies used by CPR Tutor include the following:

- **OK.**
  Reflects that the student's response matches that of the expert. This is used for identical matches as well as **nopref** conditions.
- **The operator you now need to perform is: <operator>.**
  This prompt is given in response to the user's request for help.
- **<Warning message>** presented on a cleared screen.
  A serious infraction on the user's part results in a specific warning message.

---

[7]The tutor module is implemented by the *metutor* program developed by Professor Rowe. It is contained in Appendix E.

- **<Warning message>** presented on the same screen.
  A less serious infraction on the user's part results in a specific warning message.
- **That operator requires that** <required preconditions list>.
  Indicates a mandatory precondition must first be satisfied before the operator can be applied. This violation is less severe than the above two cases.
- **That will not affect anything.**
  The action chosen by the student will have no affect on the current state nor on the overall solution.
- **That does not seem immediately helpful, but I will try it.**
  Allows the user to pursue their selected solution path.
- **I will try it, but it is not recommended or need for the problem.**
  Allows the student to apply an operator even though the system sees no need.

A central issue involved with the use of these tutoring strategies is the appropriateness of providing immediate responses to the user's action as compared with temporarily allowing the student to continue down an incorrect path without interruption. In many application areas, several solution paths may exist. The CPR procedure, however, is not loosely ordered. Much medical research has been conducted in determining the optimal ordering of operators. Although deviations from this order may not cause serious negative results, the incorrect ordering may, in fact, render the efforts of the rescuer ineffective. The CPR Tutor, therefore, is typically very stringent in the requirement that operators be performed in a specified order; students are not allowed to deviate to any great extent. It should be noted that this restrictive nature is not invoked by the tutor module itself, but rather by the CPR knowledge utilized by the tutor. When necessary to allow the student the option of choosing either of the two operators, the two argument predicate **nopref** is used. This is used in the CPR Tutor in the selection of the first two operators; represented by the Prolog fact:

nopref(check_if_unconscious,yell_for_help).

E. STUDENT MODEL

The student model is a representation of the student's understanding of CPR procedures as perceived by the CPR Tutor system. It is thus a model of the system's knowledge of the student based on their response to the specific situational state presented to the student. It is constructed by comparing the student's response to the expert module's means-ends analysis recommended response. The model is updated throughout the tutoring session.

Ingrained in the *metutor* program developed by Professor Rowe is a dynamic student model: a stack representation of the applied operators. Throughout the tutoring session, it was the current version of this stack which was used by the means-ends analysis subprogram to determine the tutoring strategy to be used. By analyzing the stack, a theory was developed as to what the student was actually trying to do. Although the human mental process isn't normally thought of in terms of stacks, it often seems like such a structure is used in the planning of complicated problems. The use of stacks in modeling the student was a subtle, yet complicated process.

A second use of the student model was much less complicated. It involved the creation of lists which captured the process used by the student. These lists were presented to the user at the completion of each tutoring session. They include a list of all the operators chosen by the user throughout the session, the operators for which the user requested help, and the operators which the user had difficulty in applying. Additionally, the student's total response time was calculated. This was kept for two purposes. One was to enforce some sense of urgency in knowing what do in each situation. The second was to present a relative overall response time measurement indicator to the user at the completion of the session. As such, diagnostics were provided to the user from which they could analyze their demonstrated misunderstandings. This self analysis approach

was used rather than a "score" because the author felt the domain of CPR, a life-or-death matter, should not be measured in terms of partial correctness. A detailed description of the specific diagnostics presented to the user are discussed later in this chapter in the user interface section.

F. USER INTERFACE

1. Input

The CPR Tutor prototype uses primarily a menu-based approach to obtain user input. The spcifice menus presented to the user are dependent on the experience level chosen, i.e., novice, intermediate, advanced.

a. Menu Advantages

Menus offer some primary advantages in their use. Firstly, an easy way is provided for the user to enter such data as user-experience level and desired-system option, i.e., HELP, Review CPR procedures, CPR skill test and QUIT. Subsequently, menus allow the user to easily interact with CPR Tutor, entering their recommended operator to the given state in the scenario.

Menus have the advantage of being especially easy to use for slow and/or inaccurate typists. There exists a lesser chance for typographical error and a resultant misrecognition by the system of the user's intended response. By being easy to use and self-explanatory by their very nature, the student's energy can be directed at learning the CPR procedure and not at how to use the system itself.

b. Menu Disadvantages

A primary disadvantage of menus is their restrictive nature. Users are not free to enter anything; their choice must be one of the menu items. The use of natural language input would seem to be a more natural and less restrictive form of communication for the user. Although natural language may ultimately provide the best

form for user input, it does not come without a cost. The use of natural languages in AI applications is a research area unto itself. Due to the inherent complexities and resource requirements of implementing a natural language input into the CPR Tutor prototype, this option was discarded for the most part. An exception is the pseudo natural language input option provided to the user of entering *help* or *quit* in lieu of a CPR operator. These options will be discussed later in this chapter.

c. Application

After the initial display which offers a Prolog reminder that all entries must be followed by a period, the user is prompted to enter their name. The main menu, displayed in Figure 4.5, is then presented, offering the user the option of requesting help, reviewing the major steps involved in providing CPR, testing their knowledge about CPR procedures, or terminating the session. Upon selecting item three of the main menu, *CPR skill test*, the major files are consulted and the user is then prompted via a menu to enter

---

fred, if you are a first time user, suggest you first request HELP

****

1. HELP
2. Review CPR procedures
3. CPR skill test
4. QUIT

ENTER YOUR CHOICE AS 1., 2., 3., or 4.

****

|: 3.

Figure 4.5 CPR Tutor Main Menu

---

their experience level. The three choices are: Novice, Intermediate, and Advanced. Subsequent menu displays are based upon this selection.

At every step in the simulation, the Novice user is presented a menu of five possible operators, one of which is always preferred. By limiting the possibilities for selection, the Novice user has a greater chance of chosing the recommended operator. The user enters the number corresponding to their choice. A validation routine is used to ensure the user's input is within the range of one to five or is either "help" or "quit" A sample Novice user menu is displayed in Figure 4.6.

The menu presented to the intermediate user is an expansion of the one presented to the novice user: fifteen possible operators are provided. Again, the recommended operator is always present in the menu. The dictating criteria for

---

POSSIBLE OPERATORS ARE:

1. check_if_unconscious
2. recheck_pulse
3. instruct_someone_to_phone_paramedic
4. position_victim
5. yell_for_help

****
The following facts are now true:
assistance is unavailable, "help" is yelled, and victim is unconscious.
What operator do you choose? 1.

That will not affect anything.

Figure 4.6 Novice User Menu

---

providing fifteen instead of all twenty-one possible operators was based on human factors: the physical size of the screen did not allow all twenty-one operators, as well as the state summation, to be viewed simultaneously. The deletion of the remaining six operators in the menu was viewed as relatively insignificant. A sample intermediate user menu is displayed in Figure 4.7.

POSSIBLE OPERATORS ARE:

1. check_if_breathing
2. check_if_unconscious
3. check_pulse
4. recheck_pulse
5. give_two_breaths
6. instruct_someone_to_phone_paramedic
7. open_airway
8. perform_emergency_breathing
9. perform_fifteen_compressions
10. perform_three_compression_breathing_cycles
11. position_victim
12. retilt_head
13. stop_bleeding
14. temporarily_stop_cpr
15. yell_for_help

****

The following facts are now true:
victim is on_back, victim is bleeding, victim is unconscious,
and assistance is available.
What operator do you choose? 13.

YOU MUST be primarily concerned with the pulse and breath of the victim!!!
You must temporarily ignore any injuries.

Figure 4.7 Intermediate User Menu

To provide a more challenging environment to the advanced user, a different approach was used. To increase the number of possible operators by an order of magnitude, operator phrases were no longer utilized in the menu. Instead, all operators are broken down into two parts The user enters their selection by forming a phrase consisting of one verb and one noun phrase. To ensure distinction between the two parts of the phrase, the menu is comprised of two distinct columns. The advanced user menu is displayed in Figure 4.8.

POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	breathing
4.	locate	d.	breaths
5.	open	e.	compression breathing cycle(s)
6.	perform	f.	compression position
7.	phone	g.	compressions
8.	position	h.	emergency breathing
9.	recheck	i.	finger sweep
10.	retilt	j.	head
11.	stop	k.	"help"
12.	standby	l.	paramedic
13.	yell	m.	someone to phone paramedic
		n.	to assist if necessary
		o.	unconsciousness

****

The following facts are now true:
victim is bleeding and victim is unconscious.
What operator do you choose? [1,c].

That operator requires that victim must be on_back and airway must be open.

Figure 4.8 Advanced User Menu

An additional menu is presented to the advanced user to obtain a specific number of repetitions for some operators. This menu is displayed in Figure 4.9.

2. Output

The primary purpose of the prototype's output was threefold. Firstly, the CPR Tutor describes the situation at every step. Each phrase in the description is a list of facts and is referred to as the **state**. This state summation is provided to the user following the

---

Your selection: give breaths

Enter the NUMBER of times the operator should be performed.

- O R -

Select from the following by entering the appropriate letter:
a. temporarily
b. forever
c. until no longer necessary, rechecking pulse every few minutes
d. 2 to 3
e. 6 to 10
f. 7 to 15

ENTER YOUR CHOICE: 2.
That operator requires that victim not breathing must be observed.

Figure 4.9 Repetition Qualifier Menu for the Advanced User

---

menu of possible operators. A natural language transcriber[8] is used to change Prolog facts into more easily readable phrases.

The second, and perhaps the most important purpose, is the system's tutorial response to the action recommended by the user. A description of these messages is in the explanation of the tutor module given above.

A third output generator is a summarizer of the user session upon successfully reaching the goal state. This summation includes a full list of user actions followed by one of more of the following:[9]

- **Operators the user had difficulty in applying.** This list is comprised of operators which were the recommended operator for a given state but were not selected by the user.
- **Operators incorrectly applied.** This list is comprised of operators which were chosen by the user as being the correct response to a given state but differed from the expert's choice.
- **Operators in which the number of repetitions was incorrectly identified by the user.** This list is only applicable to the advanced user and is discussed above.
- **Operators for which the user requested help.** This list is comprised of all operators which were the applicable response when the user entered "help" in lieu of selecting a menu item.

The summation is concluded with the user's elapsed time in completing the session. The time includes everything following the completion of the display of the menu and the input of the user's response.

---

[8]This was written by Professor Rowe of the Naval Postgraduate School and is contained in the file *metutor* contained in Appendix E.

[9]See Appendix A through C for sample session summations.

V. SYSTEM USE

A. RESOURCE REQUIREMENTS

1. System Requirements

The CPR Tutor was developed and implemented on a Digital Equipment Corporation VAX 11/780 minicomputer running the UNIX operating system. This prototype is composed of seven files: *cpr, mecpr, metutor, utilities, actions, help and quit.* Together, these files require 68K bytes of memory. Of these files, the latter three are not critical to the successful operation of the system.

2. Program Language

CPR Tutor was implemented in Prolog, a general-purpose computer programming language. The specific version utilized was C-Prolog, a Prolog interpreter written in C, developed at the University of Edinburgh [Ref. 13].

B. DEMONSTRATION

Appendix A contains a demonstration of a CPR Tutor session with a novice, first-time user. Following the suggestion of the screen prompt, the student chose to review the help file and then opted to receive an overview of the major steps involved in the CPR procedure. Appendix B contains a demonstration of a CPR Tutor session with an intermediate user. Appendix C contains a demonstration of an advanced user's session with the CPR Tutor in which the number of operator options is increased to 390.

C. LIMITATIONS

1. Realistic Simulation

An inherent limitation of the CPR Tutor system is the significant difference between its environment and that of the actual performance of CPR procedures. An important part of training is obtainable only through actual physical practice on a specially designed manikin. However, the CPR Tutor can fully test the student on their procedural knowledge. The measuring of the student's response time was designed to instill a sense of urgency, albeit a far cry from a actual human being in need of CPR to save their life.

2. Accuracy

a. Parallelism

The CPR Tutor expert module contains the knowledge base for the system. Through manipulation of the data base by the tutor module the "expert's" recommended order of CPR operators is derived. As such, the student's actions are compared to a small number of recommended solution paths. The **nopref** predicate allows two successive operators in the student's path to be applied in interchangeable order. However, it is not possible to simultaneously apply two or more operators; this restricts the system. Although not extremely affected by this restriction, the CPR process does have two operators that should be possible to perform in parallel: *check if unconscious* and *yell for help*. Application domains which require parallelism in the application of operators can not be accurately implemented.

b. Careless Errors

By using a menu-based approach in the User Interface, specifically the input portion, it is felt careless typographical errors were diminished to a considerable extent. However, when input is via a computer keyboard, typographical errors can never be totally eliminated. Validation routines were used to ensure correct format utilization

by the user and were used to ensure the user's entry was within the acceptable range. Beyond these constraints, however, there is no prevention from a student carelessly entering a wrong choice, i.e., not their actual choice, yet still an acceptable input or simply a quick input with no thought process behind it. This may require several extra responses on the student's part to return to the correct solution path and their total response time and session summary report will reflect this.

D. RECOMMENDED SYSTEM ENHANCEMENTS

1. Operator Selection

More depth in the training could be achieved by querying the user on the actual process used in applying each operator. For example, the user must locate the compression position prior to performing chest compressions. Although the user is tested on their knowledge of when to locate the position, they are not tested on actually locating the position. The further breakdown of each operator therefore represents a possibility for expanding the training capabilities of the CPR Tutor. The file *actions*, although not fully implemented, represents a start at this enhancement.

The difficulty level could easily be increased for the advanced user with the addition of more operators associated with more complex scenarios. For example, a situation could be presented in which multiple victims are present and it is first necessary for the rescuer to determine the victim in most critical need of medical attention.

2. Student Modeling

The dynamic model of the student developed by the *metutor* program utilized a stack to record the actions of the student. From an analysis of the student model, the general tutoring methodology to be used was determined. The tutor, in following a particular approach, would present to the user a message such as "That will not affect anything" as an indicator of how the system perceived the effectiveness of the student's

action. However, this approach does not attempt to reason why the student chose the operator they did. This approach would represent an enhancement.

3. Biased Randomness

The CPR Tutor currently is developed to use predefined probabilities to generate randomness. To change the assigned probabilities, it is necessary to modify the code contained the the *mecpr* file. While this is very easy to do, it is not practical for the average user. A recommended enhancement would be the use of the student model to dynamically modify these probabilities. With this enhancement, a student having difficulty with a particular area could be presented with more situations in which this defective area is relevant. As such, the student would be given more opportunities to overcome their weaknesses. Similarly, permanent storage could be used to record each user's session; information could then be gathered from previous sessions as well as the current one.

4. Graphics

The CPR Tutor strictly used a verbal "conversation" with the student. As such, the system actually related to the student the relevant facts. A more realistic approach would be to use graphics to pictorially display what is occurring. It would then be up to the student to discern the relevant facts from their own direct visual observation of the situation, the same as is necessary in real-life situations.

5. Portability

With rapidly improving technology, the possibility does exist to implement the CPR Tutor or similar system on an easily accessible microcomputer. Two possible versions of Prolog which are now available for microcomputers are Micro-Prolog and Turbo-Prolog.

VI. SUMMARY

This study was undertaken to determine the feasibility of developing and implementing an ICAI prototype system for CPR. It is not a full-scale production model. As a prototype, limitations in the system's capability were expected. The majority of these limitations were a result of the relatively limited amount of time spent in design, development and implementation of the CPR Tutor prototype. Whereas many ICAI projects are ongoing for years, by a team of researchers, the entire CPR Tutor project was completed in less than one year.

In answer to the basic research question addressed by this study, i.e., the feasibility of implementing an effective, efficient, and challenging ICAI system for the purpose of providing CPR training, the author clearly feels the answer is unequivocally "yes". In answer to the question concerning the means that should be utilized in developing such a system, the means-ends analysis tutorial approach seems effective. The CPR procedure could cleanly be decomposed into abstractions. Clear and orthogonal operators could be identified as well as the corresponding preconditions and postconditions. This is not to say, however, that other approaches might not have also been effective.

APPENDIX A - NOVICE LEVEL DEMONSTRATION

################### { Initial Screen Display } ###################

```
CCCCCC   PPPPPP   RRRRRR
CC       PP   PP  RR    RR
CC       PPPPPP   RRRRRR
CC       PP       RR RR
CCCCCC   PP       RR   RR
```

PROLOG REMINDER: ENTRIES MUST BE FOLLOWED BY A PERIOD (".")

################### { Next Screen Display } ###################

Welcome to the CPR tutor

Please enter your name: toby.

################### { Next Screen Display } ###################

toby, if you are a first time user, suggest you first request HELP

****

1. HELP
2. Review CPR procedures
3. CPR skill test
4. QUIT

ENTER YOUR CHOICE AS 1., 2., 3., or 4.

****

|: 1.

########## { Next Screen Display } ##########

help consulted 2256 bytes 0.566668 sec.

########## { Next Screen Display } ##########

C P R

One out of every two people in the United States can expect to die from a heart attack or a related disease or failure. The American Red Cross offers adult CPR courses in an effort to reduce this drastic death toll.

This CPR Tutor is designed to help maintain your level of proficiency in knowing how to provide CPR treatment at the scene of an emergency. Based upon your level of experience, you will be prompted by the tutor for your response when confronted with the current state of the emergency.

ENTER any letter key to continue: f.

########## { Next Screen Display } ##########

NOVICE

As a novice user you will be given a list of 5 possible operators, one of which is the correct response.

INTERMEDIATE

As an intermediate user you will be given a list of 15 possible operators, one of which is the correct response.

ADVANCED

As an advanced user you will be presented with two columns. Your response will be a selection of a number from column one and a letter from column two in the format "[#,letter]" such that a phrase representing the opertor will be formed.

ENTER any letter to continue: c.

##################### (Next Screen Display) ####################

HELP

At any time you are unsure of the correct response and desire help you can enter "help." or simply "h.". This will prompt the tutor to give you the preferred response.

QUIT

At the end of each session you will be provided a summary of your actions. If you desire to end a session prior to the logical completion, entering "quit." will abort the program and return you to the system prompt. No summary of the aborted session will be provided.

ENTER any letter key to return to main menu: t.

##################### (Next Screen Display) ####################

****

1. HELP
2. Review CPR procedures
3. CPR skill test
4. QUIT

ENTER YOUR CHOICE AS 1., 2., 3., or 4.

****

l. 2.

##################### (Next Screen Display) ####################

actions consulted 4692 bytes 1.05 sec.

##################### (Next Screen Display) ####################

CPR OVERVIEW - MAJOR STEPS

1. Check if unconscious.
2. Yell for help.
3. Position victim.
4. Open airway.
5. Check if breathing.
6. Give two full breaths.
7. Check pulse.
8. Instruct someone to phone paramedic - if assistance is available.
9. Locate compression position.
10. Perform 15 compressions.
11. Give two full breaths.
12. Perform 3 compression/breathing cycles.
13. Recheck pulse.
14. Temporarily cease cpr to phone paramedic - if phonecall not already made.
15. Give two full breaths.
16. Perform compression_breathing cycles until no longer applicable.

ENTER any letter key to return to main menu: d.

##################### (Next Screen Display) ####################

****

1. HELP
2. Review CPR procedures
3. CPR skill test
4. QUIT

ENTER YOUR CHOICE AS 1., 2., 3., or 4.

****

l: 3.

```
#################### ( Next Screen Display ) ####################

                    C P R  Tutor

                  INITIALIZING FILES

                   Please standby

metutor consulted 20604 bytes 4.08333 sec.
mecpr consulted 12636 bytes 3.05 sec.
actions consulted 4740 bytes 1.03334 sec.

#################### ( Next Screen Display ) ####################

                Level of experience
                -------------------
                  1. Novice
                  2. Intermediate
                  3. Advanced

Enter 1., 2., or 3. : 1.

                Good luck toby!!!

#################### ( Next Screen Display ) ####################

************************************************************
   This is a test of your CPR skills - skills that could save a life!
************************************************************

   Your objectives: victim is breathing and_or paramedic is on_scene.

      Wait a moment while I analyze the problem thoroughly.
```

##################### (Next Screen Display) ####################

POSSIBLE OPERATORS ARE:

1. check_if_unconscious
2. locate_compression_position
3. perform_three_compression_breathing_cycles
4. position_victim
5. temporarily_stop_cpr

****

The following facts are now true:
person_collapsed is true.
What operator do you choose? 1.
OK.

##################### (Next Screen Display) ####################

POSSIBLE OPERATORS ARE:

1. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
2. position_victim
3. stop_bleeding
4. temporarily_stop_cpr
5. yell_for_help

****

The following facts are now true:
victim is unconscious.
What operator do you choose? 1.
That operator requires that still_no_pulse must be true,
paramedic must be phoned, and sixth pair of breaths must be given.

#################### { Next Screen Display } ####################

POSSIBLE OPERATORS ARE:

1. check_pulse
2. open_airway
3. perform_fifteen_compressions
4. standby_to_assist
5. yell_for_help

****

The following facts are now true:
victim is unconscious.
What operator do you choose? help.
The operator you now need to perform is:
yell_for_help

#################### { Next Screen Display } ####################

POSSIBLE OPERATORS ARE:

1. check_if_breathing
2. phone_paramedic
3. retilt_head
4. standby_to_assist
5. yell_for_help

****

The following facts are now true:
victim is unconscious.
What operator do you choose? 5.
OK.

#################### (Next Screen Display) ####################

POSSIBLE OPERATORS ARE:

1. check_if_unconscious
2. recheck_pulse
3. instruct_someone_to_phone_paramedic
4. position_victim
5. yell_for_help

****

The following facts are now true:
assistance is unavailable, "help" is yelled, and victim is unconscious.
What operator do you choose? 1.
That will not affect anything.

#################### (Next Screen Display) ####################

POSSIBLE OPERATORS ARE:

1. recheck_pulse
2. perform_finger_sweep
3. perform_emergency_breathing
4. position_victim
5. standby_to_assist

****

The following facts are now true:
assistance is unavailable, "help" is yelled, and victim is unconscious.
What operator do you choose? 4.
OK.

Someone has responded to your yell for help.

```
#################### ( Next Screen Display ) ####################

POSSIBLE OPERATORS ARE:

   1. open_airway
   2. perform_fifteen_compressions
   3. standby_to_assist
   4. stop_bleeding
   5. yell_for_help

************************************************************
The following facts are now true:
assistance is available, victim is on_back, and victim is unconscious.
What operator do you choose? 1.
OK.

#################### ( Next Screen Display ) ####################

POSSIBLE OPERATORS ARE:

   1. check_if_breathing
   2. perform_abdominal_thrusts
   3. perform_emergency_breathing
   4. retilt_head
   5. temporarily_stop_cpr

************************************************************
The following facts are now true:
airway is open, assistance is available, victim is on_back,
  and victim is unconscious.
What operator do you choose? 1.
OK.
```

##################### (Next Screen Display) #####################

POSSIBLE OPERATORS ARE:

1. give_two_breaths
2. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
3. perform_three_compression_breathing_cycles
4. retilt_head
5. yell_for_help

****

The following facts are now true:
victim not breathing is observed, airway is open, assistance is available,
victim is on_back, and victim is unconscious.
What operator do you choose? 1.
OK.

##################### (Next Screen Display) #####################

POSSIBLE OPERATORS ARE:

1. check_if_unconscious
2. check_pulse
3. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
4. perform_fifteen_compressions
5. perform_three_compression_breathing_cycles

****

The following facts are now true:
first pair of breaths are given, victim not breathing is observed,
airway is open, assistance is available, victim is on_back,
and victim is unconscious.
What operator do you choose? 2.
OK.

```
#################### { Next Screen Display } ####################

POSSIBLE OPERATORS ARE:

   1. check_if_unconscious
   2. give_two_breaths
   3. locate_compression_position
   4. perform_abdominal_thrusts
   5. perform_emergency_breathing

************************************************************
The following facts are now true:
pulse is observed, first pair of breaths are given,
  victim not breathing is observed, airway is open,
  assistance is available, victim is on_back, and victim is unconscious.
What operator do you choose? 5.
OK.

The heart has stopped beating.  What should you do now?

#################### { Next Screen Display } ####################

POSSIBLE OPERATORS ARE:

   1. recheck_pulse
   2. instruct_someone_to_phone_paramedic
   3. perform_abdominal_thrusts
   4. phone_paramedic
   5. standby_to_assist

************************************************************
The following facts are now true:
victim is unconscious, no pulse is observed, first pair of breaths are given,
  victim not breathing is observed, airway is open, assistance is available,
  and victim is on_back.
What operator do you choose? 5.
That operator requires that victim must be breathing.
```

```
##################### ( Next Screen Display ) ####################

POSSIBLE OPERATORS ARE:

   1. check_if_unconscious
   2. instruct_someone_to_phone_paramedic
   3. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
   4. perform_fifteen_compressions
   5. retilt_head

************************************************************
The following facts are now true:
victim is unconscious, no pulse is observed, first pair of breaths are given,
  victim not breathing is observed, airway is open, assistance is available,
  and victim is on_back.
What operator do you choose? 2.
OK.

##################### ( Next Screen Display ) ####################

POSSIBLE OPERATORS ARE:

   1. instruct_someone_to_phone_paramedic
   2. locate_compression_position
   3. perform_three_compression_breathing_cycles
   4. standby_to_assist
   5. stop_bleeding

************************************************************
The following facts are now true:
paramedic is phoned, victim is unconscious, no pulse is observed,
  first pair of breaths are given, victim not breathing is observed,
  airway is open, and victim is on_back.
What operator do you choose? 2.
OK.
```

##################### (Next Screen Display) #####################

POSSIBLE OPERATORS ARE:

1. check_pulse
2. recheck_pulse
3. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
4. perform_emergency_breathing
5. perform_fifteen_compressions

****

The following facts are now true:
compression position is located, paramedic is phoned, victim is unconscious,
no pulse is observed, victim not breathing is observed, airway is open,
and victim is on_back.
What operator do you choose? 5.
OK.

##################### (Next Screen Display) #####################

POSSIBLE OPERATORS ARE:

1. check_if_breathing
2. give_two_breaths
3. perform_three_compression_breathing_cycles
4. position_victim
5. standby_to_assist

****

The following facts are now true:
fifteen compressions are performed, paramedic is phoned, victim is unconscious,
victim not breathing is observed, airway is open, and victim is on_back.
What operator do you choose? 2.
OK.

#################### (Next Screen Display) ####################

POSSIBLE OPERATORS ARE:

1. check_pulse
2. locate_compression_position
3. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
4. perform_fifteen_compressions
5. perform_three_compression_breathing_cycles

****

The following facts are now true.
second pair of breaths are given, fifteen compressions are performed,
paramedic is phoned, victim is unconscious, victim not breathing is observed,
airway is open, and victim is on_back.
What operator do you choose? 5.
OK.

#################### (Next Screen Display) ####################

POSSIBLE OPERATORS ARE:

1. check_if_breathing
2. check_pulse
3. recheck_pulse
4. locate_compression_position
5. perform_fifteen_compressions

****

The following facts are now true:
compression breathing cycles are performed, paramedic is phoned,
victim is unconscious, victim not breathing is observed,
airway is open, and victim is on_back.
What operator do you choose? 3.
OK.

#################### (Next Screen Display) ####################

POSSIBLE OPERATORS ARE:

1. recheck_pulse
2. give_two_breaths
3. instruct_someone_to_phone_paramedic
4. locate_compression_position
5. stop_bleeding

****

The following facts are now true:
still_no_pulse is true, paramedic is phoned, victim is unconscious,
victim not breathing is observed, airway is open, and victim is on_back.
What operator do you choose? 2.
OK.

#################### (Next Screen Display) ####################

POSSIBLE OPERATORS ARE:

1. check_pulse
2. locate_compression_position
3. perform_finger_sweep
4. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
5. standby_to_assist

****

The following facts are now true:
sixth pair of breaths are given, still_no_pulse is true, paramedic is phoned,
victim is unconscious, victim not breathing is observed, airway is open,
and victim is on_back.
What operator do you choose? 4.
OK.

##################### (Next Screen Display) #####################

Full list of your actions: [check_if_unconscious,yell_for_help,position_victim,
open_airway,check_if_breathing,give_two_breaths,check_pulse,
perform_emergency_breathing,instruct_someone_to_phone_paramedic,
locate_compression_position,perform_fifteen_compressions,give_two_breaths,
perform_three_compression_breathing_cycles,recheck_pulse,give_two_breaths,
perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes]

The operators you had difficulty in applying were:
position_victim

The operators you incorrectly applied were:
standby_to_assist

The operators for which you requested help were:
yell_for_help

Your total time for responses was 2 minutes, 41 seconds.
This equates to an average response time of 8 seconds per question.

To return to main menu, enter "c." c.

##################### (END DEMONSTRATION) #####################

APPENDIX B - INTERMEDIATE LEVEL DEMONSTRATION

########## { Initial Screen Display } ##########

```
CCCCCC   PPPPPP   RRRRRR
CC       PP   PP  RR    RR
CC       PPPPPP   RRRRRR
CC       PP       RR RR
CCCCCC   PP       RR   RR
```

PROLOG REMINDER: ENTRIES MUST BE FOLLOWED BY A PERIOD (".")

########## { Next Screen Display } ##########

Welcome to the CPR tutor

Please enter your name: corky.

########## { Next Screen Display } ##########

corky, if you are a first time user, suggest you first request HELP

****

1. HELP
2. Review CPR procedures
3. CPR skill test
4 QUIT

ENTER YOUR CHOICE AS 1., 2., 3., or 4.

****

|: 3.

```
#################### ( Next Screen Display ) ####################

                    C P R  Tutor

                  INITIALIZING FILES

                    Please standby


metutor consulted 20668 bytes 4.28333 sec.
mecpr consulted 12828 bytes 3.11667 sec.
actions consulted 4740 bytes 1.08333 sec.


#################### ( Next Screen Display ) ####################

                  Level of experience
                  -------------------
                    1. Novice
                    2. Intermediate
                    3. Advanced


Enter 1., 2., or 3. : 2.


                  Good luck corky!!!


#################### ( Next Screen Display ) ####################


************************************************************
    This is a test of your CPR skills - skills that could save a life!
************************************************************


    Your objectives: victim is breathing and_or paramedic is on_scene.


      Wait a moment while I analyze the problem thoroughly.
```

```
##################### ( Next Screen Display ) #####################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_if_unconscious
  3. check_pulse
  4. recheck_pulse
  5. instruct_someone_to_phone_paramedic
  6. open_airway
  7. perform_finger_sweep
  8. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
  9. perform_emergency_breathing
 10. perform_fifteen_compressions
 11. phone_paramedic
 12. retilt_head
 13. standby_to_assist
 14. temporarily_stop_cpr
 15. yell_for_help
************************************************************
The following facts are now true:
person_collapsed is true.
What operator do you choose? 15.
OK.

Someone has responded to your yell for help.

##################### ( Next Screen Display ) #####################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_if_unconscious
  3. check_pulse
  4. recheck_pulse
  5. locate_compression_position
  6. perform_abdominal_thrusts
  7. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
  8. perform_emergency_breathing
  9. perform_three_compression_breathing_cycles
 10. phone_paramedic
 11. position_victim
 12. retilt_head
 13. stop_bleeding
 14. temporarily_stop_cpr
 15. yell_for_help
************************************************************
The following facts are now true:
assistance is available, "help" is yelled, and person_collapsed is true.
What operator do you choose? 2.
OK.
```

```
##################### ( Next Screen Display ) #####################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_if_unconscious
  3. recheck_pulse
  4. give_two_breaths
  5. instruct_someone_to_phone_paramedic
  6. perform_abdominal_thrusts
  7. perform_finger_sweep
  8. perform_emergency_breathing
  9. perform_fifteen_compressions
 10. phone_paramedic
 11. position_victim
 12. retilt_head
 13. standby_to_assist
 14. temporarily_stop_cpr
 15. yell_for_help
*************************************************************
The following facts are now true:
victim is bleeding, victim is unconscious, assistance is available,
 and "help" is yelled.
What operator do you choose? 11.
OK.
##################### ( Next Screen Display ) #####################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_if_unconscious
  3. check_pulse
  4. recheck_pulse
  5. give_two_breaths
  6. instruct_someone_to_phone_paramedic
  7. open_airway
  8. perform_emergency_breathing
  9. perform_fifteen_compressions
 10. perform_three_compression_breathing_cycles
 11. position_victim
 12. retilt_head
 13. stop_bleeding
 14. temporarily_stop_cpr
 15. yell_for_help
*************************************************************
The following facts are now true:
victim is on_back, victim is bleeding, victim is unconscious, ,
 and assistance is available.
What operator do you choose? 13.
YOU MUST be primarily concerned with the pulse and breath of the victim!!!
You must temporarily ignore any injuries.
```

##################### { Next Screen Display } #####################

POSSIBLE OPERATORS ARE:

1. check_if_breathing
2. check_if_unconscious
3. check_pulse
4. recheck_pulse
5. give_two_breaths
6. open_airway
7. perform_abdominal_thrusts
8. perform_finger_sweep
9. perform_fifteen_compressions
10. perform_three_compression_breathing_cycles
11. position_victim
12. retilt_head
13. standby_to_assist
14. temporarily_stop_cpr
15. yell_for_help

****

The following facts are now true:
victim is on_back, victim is bleeding, victim is unconscious,
and assistance is available.
What operator do you choose? 9.

##################### { Next Screen Display } #####################

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

It is important that you check the cartoid pulse for 5 to 10 seconds before starting CPR.

IT IS DANGEROUS TO PERFORM CHEST COMPRESSIONS IF THE HEART IS BEATING.

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

```
##################### { Next Screen Display } ######################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_pulse
  3. recheck_pulse
  4. give_two_breaths
  5. instruct_someone_to_phone_paramedic
  6. locate_compression_position
  7. open_airway
  8. perform_finger_sweep
  9. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
 10. perform_fifteen_compressions
 11. perform_three_compression_breathing_cycles
 12. phone_paramedic
 13. standby_to_assist
 14. stop_bleeding
 15. yell_for_help
**************************************************************
The following facts are now true:
victim is on_back, victim is bleeding, victim is unconscious,
  and assistance is available.
What operator do you choose? 7.
OK.
##################### { Next Screen Display } ######################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_if_unconscious
  3. check_pulse
  4. give_two_breaths
  5. instruct_someone_to_phone_paramedic
  6. locate_compression_position
  7. open_airway
  8. perform_abdominal_thrusts
  9. perform_finger_sweep
 10. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
 11. perform_emergency_breathing
 12. perform_three_compression_breathing_cycles
 13. retilt_head
 14. temporarily_stop_cpr
 15. yell_for_help
**************************************************************
The following facts are now true:
airway is open, victim is on_back, victim is bleeding, victim is unconscious,
  and assistance is available.
What operator do you choose? 1.
OK.
```

```
#################### ( Next Screen Display ) ####################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_if_unconscious
  3. give_two_breaths
  4. instruct_someone_to_phone_paramedic
  5. locate_compression_position
  6. open_airway
  7. perform_finger_sweep
  8. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
  9. perform_emergency_breathing
 10. perform_fifteen_compressions
 11. perform_three_compression_breathing_cycles
 12. phone_paramedic
 13. position_victim
 14. standby_to_assist
 15. temporarily_stop_cpr
************************************************************
The following facts are now true:
victim not breathing is observed, airway is open, victim is on_back,
  victim is bleeding, victim is unconscious, and assistance is available.
What operator do you choose? 3.
OK.
Airway is obstructed - you are unable to breathe air into the victim.
#################### ( Next Screen Display ) ####################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_if_unconscious
  3. check_pulse
  4. give_two_breaths
  5. instruct_someone_to_phone_paramedic
  6. locate_compression_position
  7. open_airway
  8. perform_abdominal_thrusts
  9. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
 10. perform_emergency_breathing
 11. phone_paramedic
 12. position_victim
 13. retilt_head
 14. standby_to_assist
 15. yell_for_help
************************************************************
The following facts are now true:
airway is obstructed, victim not breathing is observed, airway is open,
  victim is on_back, victim is bleeding, victim is unconscious,
  and assistance is available.
What operator do you choose? 13.
OK.
```

```
##################### { Next Screen Display } #####################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_if_unconscious
  3. check_pulse
  4. recheck_pulse
  5. give_two_breaths
  6. locate_compression_position
  7. open_airway
  8. perform_abdominal_thrusts
  9. perform_finger_sweep
 10. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
 11. perform_emergency_breathing
 12. perform_fifteen_compressions
 13. phone_paramedic
 14. position_victim
 15. standby_to_assist
*************************************************************
The following facts are now true:
head is retilt, victim not breathing is observed, airway is open,
  victim is on_back, victim is bleeding, victim is unconscious,
  and assistance is available.
What operator do you choose? 5.
OK.
##################### { Next Screen Display } #####################
POSSIBLE OPERATORS ARE:
  1. check_if_unconscious
  2. check_pulse
  3. recheck_pulse
  4. instruct_someone_to_phone_paramedic
  5. open_airway
  6. perform_abdominal_thrusts
  7. perform_finger_sweep
  8. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
  9. perform_emergency_breathing
 10. perform_fifteen_compressions
 11. perform_three_compression_breathing_cycles
 12. phone_paramedic
 13. position_victim
 14. retilt_head
 15. temporarily_stop_cpr
*************************************************************
The following facts are now true:
first pair of breaths are given, head is retilt,
  victim not breathing is observed, airway is open, victim is on_back,
  victim is bleeding, victim is unconscious, and assistance is available.
What operator do you choose? 6.
OK.
```

```
##################### ( Next Screen Display ) #####################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_pulse
  3. recheck_pulse
  4. locate_compression_position
  5. open_airway
  6. perform_abdominal_thrusts
  7. perform_finger_sweep
  8. perform_emergency_breathing
  9. perform_fifteen_compressions
 10. phone_paramedic
 11. position_victim
 12. retilt_head
 13. stop_bleeding
 14. temporarily_stop_cpr
 15. yell_for_help
*************************************************************
The following facts are now true:
abdominal thrusts are performed, airway is obstructed, first pair of breaths are given,
  head is retilt, victim not breathing is observed, airway is open, victim is on_back,
  victim is bleeding, victim is unconscious, and assistance is available.
What operator do you choose? 7.
OK.
##################### ( Next Screen Display ) #####################
POSSIBLE OPERATORS ARE:
  1 check_if_breathing
  2. check_if_unconscious
  3. check_pulse
  4. give_two_breaths
  5. instruct_someone_to_phone_paramedic
  6. locate_compression_position
  7. open_airway
  8. perform_finger_sweep
  9. perform_emergency_breathing
 10. perform_fifteen_compressions
 11. phone_paramedic
 12. position_victim
 13. retilt_head
 14. standby_to_assist
 15. stop_bleeding
*************************************************************
The following facts are now true:
finger sweep is performed, victim not breathing is observed, airway is open,
  victim is on_back, victim is bleeding, victim is unconscious, and assistance is available.
What operator do you choose? 4.
OK.
```

```
##################### ( Next Screen Display ) #####################
POSSIBLE OPERATORS ARE:
   1. check_if_breathing
   2. check_pulse
   3. recheck_pulse
   4. give_two_breaths
   5. instruct_someone_to_phone_paramedic
   6. locate_compression_position
   7. open_airway
   8. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
   9. perform_emergency_breathing
  10. perform_fifteen_compressions
  11. perform_three_compression_breathing_cycles
  12. phone_paramedic
  13. retilt_head
  14. standby_to_assist
  15. yell_for_help
*****************************************************************
The following facts are now true:
first pair of breaths are given, victim not breathing is observed,
  airway is open, victim is on_back, victim is bleeding,
  victim is unconscious, and assistance is available.
What operator do you choose? 2.
OK.
##################### ( Next Screen Display ) #####################
POSSIBLE OPERATORS ARE:
   1. check_if_breathing
   2. check_if_unconscious
   3. check_pulse
   4. recheck_pulse
   5. instruct_someone_to_phone_paramedic
   6. open_airway
   7. perform_abdominal_thrusts
   8. perform_finger_sweep
   9. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
  10. perform_emergency_breathing
  11. perform_fifteen_compressions
  12. perform_three_compression_breathing_cycles
  13. phone_paramedic
  14. position_victim
  15. stop_bleeding
*****************************************************************
The following facts are now true:
no pulse is observed, first pair of breaths are given,
  victim not breathing is observed, airway is open, victim is on_back,
  victim is bleeding, victim is unconscious, and assistance is available.
What operator do you choose? 2.
That will not affect anything.
```

```
##################### { Next Screen Display } ####################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_if_unconscious
  3. check_pulse
  4. recheck_pulse
  5. instruct_someone_to_phone_paramedic
  6. locate_compression_position
  7. open_airway
  8. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
  9. perform_emergency_breathing
 10. perform_three_compression_breathing_cycles
 11. position_victim
 12. retilt_head
 13. stop_bleeding
 14. temporarily_stop_cpr
 15. yell_for_help
************************************************************
The following facts are now true:
no pulse is observed, first pair of breaths are given,
  victim not breathing is observed, airway is open, victim is on_back,
  victim is bleeding, victim is unconscious, and assistance is available.
What operator do you choose? h.
The operator you now need to perform is: instruct_someone_to_phone_paramedic
##################### { Next Screen Display } ####################
POSSIBLE OPERATORS ARE:
  1. check_if_unconscious
  2. check_pulse
  3. recheck_pulse
  4. give_two_breaths
  5. instruct_someone_to_phone_paramedic
  6. perform_abdominal_thrusts
  7. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
  8. perform_emergency_breathing
  9. perform_fifteen_compressions
 10. perform_three_compression_breathing_cycles
 11. phone_paramedic
 12. position_victim
 13. retilt_head
 14. stop_bleeding
 15. yell_for_help
************************************************************
The following facts are now true:
no pulse is observed, first pair of breaths are given, victim not breathing is observed,
  airway is open, victim is on_back, victim is bleeding, victim is unconscious,
  and assistance is available.
What operator do you choose? 5.
OK.
```

```
#################### ( Next Screen Display ) ####################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. recheck_pulse
  3. give_two_breaths
  4. instruct_someone_to_phone_paramedic
  5. locate_compression_position
  6. open_airway
  7. perform_abdominal_thrusts
  8. perform_finger_sweep
  9. perform_emergency_breathing
 10. phone_paramedic
 11. position_victim
 12. retilt_head
 13. standby_to_assist
 14. stop_bleeding
 15. yell_for_help
************************************************************
The following facts are now true:
paramedic is phoned, no pulse is observed, first pair of breaths are given,
  victim not breathing is observed, airway is open, victim is on_back,
  victim is bleeding, and victim is unconscious.
What operator do you choose? 10.
That operator requires that still_no_pulse must be true,
 assistance must be unavailable, and cpr must be temporarily_stopped.
#################### ( Next Screen Display ) ####################
POSSIBLE OPERATORS ARE:
  1. recheck_pulse
  2. give_two_breaths
  3. instruct_someone_to_phone_paramedic
  4. locate_compression_position
  5. open_airway
  6. perform_abdominal_thrusts
  7. perform_finger_sweep
  8. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
  9. perform_emergency_breathing
 10. perform_fifteen_compressions
 11. perform_three_compression_breathing_cycles
 12. phone_paramedic
 13. standby_to_assist
 14. temporarily_stop_cpr
 15. yell_for_help
************************************************************
The following facts are now true:
paramedic is phoned, no pulse is observed, first pair of breaths are given,
      victim is bleeding, victim not breathing is observed, airway is open,
      victim is on_back, and victim is unconscious.
What operator do you choose? 4.       OK.
```

```
##################### ( Next Screen Display ) ######################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_if_unconscious
  3. give_two_breaths
  4. instruct_someone_to_phone_paramedic
  5. locate_compression_position
  6. perform_abdominal_thrusts
  7. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
  8. perform_emergency_breathing
  9. perform_fifteen_compressions
 10. perform_three_compression_breathing_cycles
 11. phone_paramedic
 12. position_victim
 13. retilt_head
 14. standby_to_assist
 15. stop_bleeding
**************************************************************
The following facts are now true:
compression position is located, paramedic is phoned, no pulse is observed,
  victim not breathing is observed, airway is open, victim is on_back,
  victim is bleeding, and victim is unconscious.
What operator do you choose? 9.
OK.
##################### ( Next Screen Display ) ######################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_pulse
  3. recheck_pulse
  4. give_two_breaths
  5. instruct_someone_to_phone_paramedic
  6. open_airway
  7. perform_abdominal_thrusts
  8. perform_finger_sweep
  9. perform_emergency_breathing
 10. perform_three_compression_breathing_cycles
 11. position_victim
 12. retilt_head
 13. standby_to_assist
 14. stop_bleeding
 15. temporarily_stop_cpr
**************************************************************
The following facts are now true:
fifteen compressions are performed, paramedic is phoned,
  victim not breathing is observed, airway is open, victim is on_back,
  victim is bleeding, and victim is unconscious.
What operator do you choose? 4.
OK.
```

```
##################### ( Next Screen Display ) ######################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_if_unconscious
  3. give_two_breaths
  4. instruct_someone_to_phone_paramedic
  5. locate_compression_position
  6. open_airway
  7. perform_abdominal_thrusts
  8. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
  9. perform_emergency_breathing
 10. perform_three_compression_breathing_cycles
 11. retilt_head
 12. standby_to_assist
 13. stop_bleeding
 14. temporarily_stop_cpr
 15. yell_for_help
************************************************************
The following facts are now true:
second pair of breaths are given, fifteen compressions are performed,
  paramedic is phoned, victim not breathing is observed, airway is open,
  victim is on_back, victim is bleeding, and victim is unconscious.
What operator do you choose? 10
OK.
##################### ( Next Screen Display ) ######################
POSSIBLE OPERATORS ARE:
  1. check_if_breathing
  2. check_if_unconscious
  3. recheck_pulse
  4. give_two_breaths
  5. instruct_someone_to_phone_paramedic
  6. locate_compression_position
  7. open_airway
  8. perform_abdominal_thrusts
  9. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
 10. perform_fifteen_compressions
 11. phone_paramedic
 12. retilt_head
 13. standby_to_assist
 14. stop_bleeding
 15. yell_for_help
************************************************************
The following facts are now true:
compression breathing cycles are performed, paramedic is phoned,
  victim not breathing is observed, airway is open, victim is on_back,
  victim is bleeding, and victim is unconscious.
What operator do you choose? 3.
OK.
```

```
##################### ( Next Screen Display ) ####################
POSSIBLE OPERATORS ARE:
  1. check_if_unconscious
  2. check_pulse
  3. give_two_breaths
  4. open_airway
  5. perform_abdominal_thrusts
  6. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
  7. perform_emergency_breathing
  8. perform_three_compression_breathing_cycles
  9. phone_paramedic
 10. position_victim
 11. retilt_head
 12. standby_to_assist
 13. stop_bleeding
 14. temporarily_stop_cpr
 15. yell_for_help
*************************************************************
The following facts are now true:
still_no_pulse is true, paramedic is phoned, victim not breathing is observed,
  airway is open, victim is on_back, victim is bleeding,
  and victim is unconscious.
What operator do you choose? 14.
That operator requires that assistance must be unavailable.
##################### ( Next Screen Display ) ####################
POSSIBLE OPERATORS ARE:
  1. check_if_unconscious
  2. check_pulse
  3. recheck_pulse
  4. give_two_breaths
  5. instruct_someone_to_phone_paramedic
  6. locate_compression_position
  7. open_airway
  8. perform_abdominal_thrusts
  9. perform_finger_sweep
 10. perform_emergency_breathing
 11. perform_three_compression_breathing_cycles
 12. position_victim
 13. retilt_head
 14. stop_bleeding
 15. temporarily_stop_cpr
*************************************************************
The following facts are now true:
still_no_pulse is true, paramedic is phoned, victim not breathing is observed,
  airway is open, victim is on_back, victim is bleeding,
  and victim is unconscious.
What operator do you choose? 4.
OK.
```

```
##################### ( Next Screen Display ) #####################
POSSIBLE OPERATORS ARE:
  1. check_if_unconscious
  2. recheck_pulse
  3. give_two_breaths
  4. locate_compression_position
  5. open_airway
  6. perform_abdominal_thrusts
  7. perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes
  8. perform_emergency_breathing
  9. perform_fifteen_compressions
 10. perform_three_compression_breathing_cycles
 11. phone_paramedic
 12. position_victim
 13. standby_to_assist
 14. stop_bleeding
 15. yell_for_help
**************************************************************************
The following facts are now true:
sixth pair of breaths are given, still_no_pulse is true, paramedic is phoned,
  victim not breathing is observed, airway is open, victim is on_back,
  victim is bleeding, and victim is unconscious.
What operator do you choose? 7.
OK.
##################### ( Next Screen Display ) #####################
Full list of your actions: [yell_for_help,check_if_unconscious,position_victim,
  open_airway,check_if_breathing,give_two_breaths,retilt_head,give_two_breaths,
  perform_abdominal_thrusts,perform_finger_sweep,give_two_breaths,check_pulse,
  instruct_someone_to_phone_paramedic,locate_compression_position,
  perform_fifteen_compressions,give_two_breaths,
  perform_three_compression_breathing_cycles,recheck_pulse,give_two_breaths,
  perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes]

The operators you had difficulty in applying were:
  instruct_someone_to_phone_paramedic

The operators you incorrectly applied were:
  temporarily_stop_cpr

The operators for which you requested help were:
  instruct_someone_to_phone_paramedic

Your total time for responses was 6 minutes, 16 seconds.
This equates to an average response time of 14 seconds per question.
--------------------------------------------------------------------------
To return to main menu, enter "c." c.

##################### ( END DEMONSTRATION ) #####################
```

APPENDIX C - ADVANCED LEVEL DEMONSTRATION

################## { Initial Screen Display } ##################

```
CCCCCC   PPPPPP   RRRRRR
CC       PP   PP  RR    RR
CC       PPPPPP   RRRRRR
CC       PP       RR RR
CCCCCC   PP       RR   RR
```

PROLOG REMINDER: ENTRIES MUST BE FOLLOWED BY A PERIOD (".")

################## { Next Screen Display } ##################

Welcome to the CPR tutor

Please enter your name: fred.

################## { Next Screen Display } ##################

fred, if you are a first time user, suggest you first request HELP

****

1. HELP
2. Review CPR procedures
3. CPR skill test
4. QUIT

ENTER YOUR CHOICE AS 1., 2., 3., or 4.

****

|: 3.

################## (Next Screen Display) ##################

C P R Tutor

INITIALIZING FILES

Please standby

metutor consulted 20680 bytes 4.1 sec.
mecpr consulted 12800 bytes 3.11667 sec.
actions consulted 4692 bytes 1.06667 sec.

################## (Next Screen Display) ##################

Level of experience
---
1. Novice
2. Intermediate
3. Advanced

Enter 1., 2., or 3. : 3.

Good luck fred!!!

################## (Next Screen Display) ##################

****
This is a test of your CPR skills - skills that could save a life!
****

Your objectives: victim is breathing and_or paramedic is on_scene.

Wait a moment while I analyze the problem thoroughly.

################## { Next Screen Display } ##################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	bleeding
4.	locate	d.	breathing
5.	open	e.	breaths
6.	perform	f.	compression position
7.	phone	g.	compressions
8.	position	h.	cpr
9.	recheck	i.	emergency breathing
10.	retilt	j.	finger sweep
11.	stop	k.	paramedic
12.	standby	l.	pulse
13.	yell	m.	someone to phone paramedic
		n.	unconsciousness
		o.	victim

ENTER YOUR CHOICE BY SELECTING ONE ITEM FROM EACH COLUMN
Example: "[2,k]."
****
The following facts are now true: person_collapsed is true.
What operator do you choose? [1].
Input must be in the form [number,letter] and within specified range
Please reenter: [1,n]. OK.
################## { Next Screen Display } ##################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	bleeding
4.	locate	d.	breathing
5.	open	e.	compression breathing cycle(s)
6.	perform	f.	compression position
7.	phone	g.	compressions
8.	position	h.	emergency breathing
9.	recheck	i.	head
10.	retilt	j.	"help"
11.	stop	k.	paramedic
12.	standby	l.	pulse
13.	yell	m.	someone to phone paramedic
		n.	unconsciousness
		o.	victim

****
The following facts are now true: victim is bleeding and victim is unconscious.
What operator do you choose? [11,c].
YOU MUST be primarily concerned with the pulse and breath of the victim!!!
You must temporarily ignore any injuries.

##################### { Next Screen Display } ####################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	breathing
4.	locate	d.	breaths
5.	open	e.	compression breathing cycle(s)
6.	perform	f.	compression position
7.	phone	g.	compressions
8.	position	h.	emergency breathing
9.	recheck	i.	finger sweep
10.	retilt	j.	head
11.	stop	k.	"help"
12.	standby	l.	paramedic
13.	yell	m.	someone to phone paramedic
		n.	to assist if necessary
		o.	unconsciousness

****

The following facts are now true:
victim is bleeding and victim is unconscious.
What operator do you choose? [1,c].
That operator requires that victim must be on_back and airway must be open.
##################### { Next Screen Display } ####################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	bleeding
4.	locate	d.	breathing
5.	open	e.	breaths
6.	perform	f.	compression breathing cycle(s)
7.	phone	g.	compression position
8.	position	h.	compressions
9.	recheck	i.	emergency breathing
10.	retilt	j.	"help"
11.	stop	k.	paramedic
12.	standby	l.	pulse
13.	yell	m.	someone to phone paramedic
		n.	to assist if necessary
		o.	victim

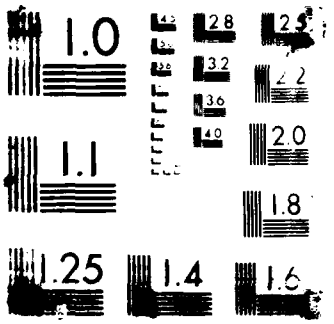
****

The following facts are now true:
victim is bleeding and victim is unconscious.
What operator do you choose? [5 d].
Your input is not an valid operator. Please try again.

AD-A198 554 AN INTELLIGENT COMPUTER-ASSISTED INSTRUCTION SYSTEM FOR CARDIOPULMONARY RESUSCITATION(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA D S CAMPBELL JUN 88 2/2

UNCLASSIFIED F/G 12/9 NL

END
DATE FILMED
10 88


1.0
1.1
1.25
1.4
1.6
2.0
1.8
2.2
2.5
2.8
3.2
3.6
4.0

###################### (Next Screen Display) ######################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	breathing
4.	locate	d.	breaths
5.	open	e.	compression breathing cycle(s)
6.	perform	f.	compression position
7.	phone	g.	compressions
8.	position	h.	cpr
9.	recheck	i.	emergency breathing
10.	retilt	j.	head
11.	stop	k.	"help"
12.	standby	l.	someone to phone paramedic
13.	yell	m.	to assist if necessary
		n.	unconsciousness
		o.	victim

****

The following facts are true:
victim is bleeding and victim is unconscious.
What operator do you choose?[13,k].
OK.
Someone has responded to your yell for help.
###################### (Next Screen Display) ######################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	bleeding
4.	locate	d.	breathing
5.	open	e.	breaths
6.	perform	f.	compressions
7.	phone	g.	cpr
8.	position	h.	emergency breathing
9.	recheck	i.	finger sweep
10.	retilt	j.	"help"
11.	stop	k.	pulse
12.	standby	l.	someone to phone paramedic
13.	yell	m.	to assist if necessary
		n.	unconsciousness
		o.	victim

****

The following facts are now true:
assistance is available, "help" is yelled, victim is bleeding, and victim is unconscious.
What operator do you choose? [8,o].
OK.

###################### (Next Screen Display) ######################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	airway
2.	give	b.	bleeding
3.	instruct	c.	breathing
4.	locate	d.	breaths
5.	open	e.	compression breathing cycle(s)
6.	perform	f.	compression position
7.	phone	g.	compressions
8.	position	h.	cpr
9.	recheck	i.	emergency breathing
10.	retilt	j.	"help"
11.	stop	k.	paramedic
12.	standby	l.	pulse
13.	yell	m	someone to phone paramedic
		n.	unconsciousness
		o.	victim

****

The following facts are now true:
victim is on_back, assistance is available, victim is bleeding, and victim is unconscious.
What operator do you choose? [2,d].

###################### (Next Screen Display) ######################

Your selection: give breaths

Enter the NUMBER of times the operator should be performed.

- O R -

Select from the following by entering the appropriate letter:

a. temporarily
b. forever
c. until no longer necessary, rechecking pulse every few minutes
d. 2 to 3
e. 6 to 10
f. 7 to 15

ENTER YOUR CHOICE: 2.
That operator requires that victim not breathing must be observed.

#################### (Next Screen Display) ####################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	airway
2.	give	b.	bleeding
3.	instruct	c.	compression breathing cycle(s)
4.	locate	d.	compression position
5.	open	e.	compressions
6.	perform	f.	cpr
7.	phone	g.	emergency breathing
8.	position	h.	finger sweep
9.	recheck	i.	head
10.	retilt	j.	"help"
11.	stop	k.	paramedic
12.	standby	l.	someone to phone paramedic
13.	yell	m.	to assist if necessary
		n.	unconsciousness
		o.	victim

****

The following facts are now true:
victim is on_back, assistance is available, victim is bleeding, and victim is unconscious.
What operator do you choose? h.
The operator you now need to perform is: open_airway
#################### (Next Screen Display) ####################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	bleeding
4.	locate	d.	breathing
5.	open	e.	breaths
6.	perform	f.	compression breathing cycle(s)
7.	phone	g.	cpr
8.	position	h.	emergency breathing
9.	recheck	i.	head
10.	retilt	j.	"help"
11.	stop	k.	paramedic
12.	standby	l.	someone to phone paramedic
13.	yell	m.	to assist if necessary
		n.	unconsciousness
		o.	victim

****

The following facts are now true:
victim is on_back, assistance is available, victim is bleeding, and victim is unconscious.
What operator do you choose? [5,b].
OK.

###################### (Next Screen Display) ######################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	bleeding
3.	instruct	c.	breathing
4.	locate	d.	breaths
5.	open	e.	compression breathing cycle(s)
6.	perform	f.	compression position
7.	phone	g.	compressions
8.	position	h.	cpr
9.	recheck	i.	finger sweep
10.	retilt	j.	"help"
11.	stop	k.	paramedic
12.	standby	l.	pulse
13.	yell	m.	someone to phone paramedic
		n.	to assist if necessary
		o.	unconsciousness

****

The following facts are now true:
airway is open, victim is on_back, assistance is available, victim is bleeding, and victim is unconscious.
What operator do you choose? [1,c].
OK.
###################### (Next Screen Display) ######################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	airway
2.	give	b.	bleeding
3.	instruct	c.	breathing
4.	locate	d.	breaths
5.	open	e.	compression breathing cycle(s)
6.	perform	f.	compression position
7.	phone	g.	compressions
8.	position	h.	cpr
9.	recheck	i.	emergency breathing
10.	retilt	j.	finger sweep
11.	stop	k.	paramedic
12.	standby	l.	pulse
13.	yell	m.	to assist if necessary
		n.	unconsciousness
		o.	victim

****

The following facts are now true:
victim is breathing, airway is open, victim is on_back, assistance is available, victim is bleeding, and victim is unconscious.
What operator do you choose? [12,m].
That operator requires that victim must not be bleeding.

##################### (Next Screen Display) ####################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1. check	a. abdominal thrusts
2. give	b. bleeding
3. instruct	c. breaths
4. locate	d. compression breathing cycle(s)
5. open	e. compression position
6. perform	f. compressions
7. phone	g. cpr
8. position	h. emergency breathing
9. recheck	i. finger sweep
10. retilt	j. head
11. stop	k. "help"
12. standby	l. paramedic
13. yell	m. pulse
	n. someone to phone paramedic
	o. victim

****

The following facts are now true:
victim is breathing, airway is open, victim is on_back, assistance is available,
victim is bleeding, and victim is unconscious.
What operator do you choose? [11,b].
OK.
##################### (Next Screen Display) ####################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1. check	a. abdominal thrusts
2. give	b. airway
3. instruct	c. breathing
4. locate	d. breaths
5. open	e. compression position
6. perform	f. compressions
7. phone	g. cpr
8. position	h. emergency breathing
9. recheck	i. "help"
10. retilt	j. paramedic
11. stop	k. pulse
12. standby	l. someone to phone paramedic
13. yell	m. to assist if necessary
	n. unconsciousness
	o. victim

****

The following facts are now true:
bleeding is stopped, victim is breathing, airway is open, victim is on_back,
assistance is available, and victim is unconscious.
What operator do you choose? [12,m].
OK.
Breathing has stopped. What should you do now?

##################### (Next Screen Display) ####################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	breathing
4.	locate	d.	breaths
5.	open	e.	compression breathing cycle(s)
6.	perform	f.	compression position
7.	phone	g.	compressions
8.	position	h.	cpr
9.	recheck	i.	finger sweep
10.	retilt	j.	head
11.	stop	k.	"help"
12.	standby	l.	paramedic
13.	yell	m.	someone to phone paramedic
		n.	to assist if necessary
		o.	victim

****
The following facts are now true:
victim not breathing is observed, stopped_breathing is true, airway is open,
victim is on_back, assistance is available, and victim is unconscious.
What operator do you choose? [2,d].
##################### (Next Screen Display) ####################

Your selection: give breaths

Enter the NUMBER of times the operator should be performed.

- O R -

Select from the following by entering the appropriate letter:

a. temporarily
b. forever
c. until no longer necessary, rechecking pulse every few minutes
d. 2 to 3
e. 6 to 10
f. 7 to 15

ENTER YOUR CHOICE: a.
temporarily is not correct.
Try again: 4.
give breaths is a valid operator.
It should be done 2 times.

****
OK.

###################### { Next Screen Display } ######################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	bleeding
4.	locate	d.	compression breathing cycle(s)
5.	open	e.	compression position
6.	perform	f.	compressions
7.	phone	g.	cpr
8.	position	h.	emergency breathing
9.	recheck	i.	finger sweep
10.	retilt	j.	"help"
11.	stop	k.	paramedic
12.	standby	l.	pulse
13.	yell	m.	to assist if necessary
		n.	unconsciousness
		o.	victim

****

The following facts are now true:
first pair of breaths are given, victim not breathing is observed, airway is open, victim is on_back, assistance is available, and victim is unconscious.
What operator do you choose? [6,f].
###################### { Next Screen Display } ######################

Your selection: perform compressions

Enter the NUMBER of times the operator should be performed.

- O R -

Select from the following by entering the appropriate letter:
a. temporarily
b. forever
c. until no longer necessary, rechecking pulse every few minutes
d. 2 to 3
e. 6 to 10
f. 7 to 15

ENTER YOUR CHOICE: 15.

#################### (Next Screen Display) ####################

@@

It is important that you check the cartoid pulse for 5 to 10 seconds before starting CPR.

IT IS DANGEROUS TO PERFORM CHEST COMPRESSIONS IF THE HEART IS BEATING.

@@

#################### (Next Screen Display) ####################

POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	bleeding
3.	instruct	c.	breathing
4.	locate	d.	breaths
5.	open	e.	compression breathing cycle(s)
6.	perform	f.	compression position
7.	phone	g.	compressions
8.	position	h.	cpr
9.	recheck	i.	emergency breathing
10.	retilt	j.	finger sweep
11.	stop	k.	"help"
12.	standby	l.	paramedic
13.	yell	m.	pulse
		n.	to assist if necessary
		o.	unconsciousness

****

The following facts are now true:
first pair of breaths are given, victim not breathing is observed, airway is open, victim is on_back, assistance is available, and victim is unconscious.
What operator do you choose? [1,m].
OK.

##################### (Next Screen Display) ####################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	bleeding
4.	locate	d.	breathing
5.	open	e.	breaths
6.	perform	f.	compression breathing cycle(s)
7.	phone	g.	compression position
8.	position	h.	compressions
9.	recheck	i.	finger sweep
10.	retilt	j.	head
11.	stop	k.	"help"
12.	standby	l.	someone to phone paramedic
13.	yell	m.	to assist if necessary
		n.	unconsciousness
		o.	victim

****

The following facts are now true:
no pulse is observed, first pair of breaths are given, victim not breathing is observed,
airway is open, victim is on_back, assistance is available, and victim is unconscious.
What operator do you choose? [3,l].
OK.

##################### (Next Screen Display) ####################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	airway
2.	give	b.	breathing
3.	instruct	c.	compression breathing cycle(s)
4.	locate	d.	compression position
5.	open	e.	compressions
6.	perform	f.	cpr
7.	phone	g.	emergency breathing
8.	position	h.	finger sweep
9.	recheck	i.	head
10.	retilt	j.	"help"
11.	stop	k.	paramedic
12.	standby	l.	pulse
13.	yell	m.	someone to phone paramedic
		n.	to assist if necessary
		o.	victim

****

The following facts are now true:
paramedic is phoned, no pulse is observed, first pair of breaths are given,
victim not breathing is observed, airway is open, victim is on_back,
and victim is unconscious.
What operator do you choose? [4,d].
OK.

###################### (Next Screen Display) ######################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	bleeding
4.	locate	d.	breathing
5.	open	e.	compression breathing cycle(s)
6.	perform	f.	compressions
7.	phone	g.	cpr
8.	position	h.	emergency breathing
9.	recheck	i.	finger sweep
10.	retilt	j.	head
11.	stop	k.	"help"
12.	standby	l.	paramedic
13.	yell	m.	pulse
		n.	someone to phone paramedic
		o.	unconsciousness

****

The following facts are now true:
compression position is located, paramedic is phoned, no pulse is observed,
victim not breathing is observed, airway is open, victim is on_back,
and victim is unconscious.
What operator do you choose? [2,f].
###################### (Next Screen Display) ######################

Your selection: give compressions

Enter the NUMBER of times the operator should be performed.

- O R -

Select from the following by entering the appropriate letter:

a. temporarily
b. forever
c. until no longer necessary, rechecking pulse every few minutes
d. 2 to 3
e. 6 to 10
f. 7 to 15

ENTER YOUR CHOICE: f.
7 to 15 is not correct.
Try again: 15.
OK.

########## (Next Screen Display) ##########
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	bleeding
4.	locate	d.	breaths
5.	open	e.	compression position
6.	perform	f.	compressions
7.	phone	g.	cpr
8.	position	h.	finger sweep
9.	recheck	i.	"help"
10.	retilt	j.	paramedic
11	stop	k.	pulse
12.	standby	l.	someone to phone paramedic
13.	yell	m.	to assist if necessary
		n.	unconsciousness
		o.	victim

****

The following facts are now true:
fifteen compressions are performed, paramedic is phoned, victim not breathing is observed, airway is open, victim is on_back, and victim is unconscious.
What operator do you choose? [2,d].

########## (Next Screen Display) ##########

Your selection: give breaths

Enter the NUMBER of times the operator should be performed.

- O R -

Select from the following by entering the appropriate letter:

a. temporarily
b. forever
c. until no longer necessary, rechecking pulse every few minutes
d. 2 to 3
e. 6 to 10
f. 7 to 15

ENTER YOUR CHOICE: 2.
OK.

########## (Next Screen Display) ##########
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	bleeding
4.	locate	d.	breathing
5.	open	e.	breaths
6.	perform	f.	compression breathing cycle(s)
7.	phone	g.	cpr
8.	position	h.	emergency breathing
9.	recheck	i.	finger sweep
10.	retilt	j.	head
11.	stop	k.	"help"
12.	standby	l.	paramedic
13.	yell	m.	to assist if necessary
		n.	unconsciousness
		o.	victim

****

The following facts are now true:
second pair of breaths are given, fifteen compressions are performed,
paramedic is phoned, victim not breathing is observed, airway is open,
victim is on_back, and victim is unconscious.
What operator do you choose? [6,f].
########## (Next Screen Display) ##########

Your selection: perform compression breathing cycle(s)

Enter the NUMBER of times the operator should be performed.

- O R -

Select from the following by entering the appropriate letter:
a. temporarily
b. forever
c. until no longer necessary, rechecking pulse every few minutes
d. 2 to 3
e. 6 to 10
f. 7 to 15

ENTER YOUR CHOICE: 3.
OK.

#################### (Next Screen Display) ####################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	airway
2.	give	b.	bleeding
3.	instruct	c.	breathing
4.	locate	d.	breaths
5.	open	e.	compression breathing cycle(s)
6.	perform	f.	compression position
7.	phone	g.	cpr
8.	position	h.	emergency breathing
9.	recheck	i.	finger sweep
10.	retilt	j.	"help"
11.	stop	k.	pulse
12.	standby	l.	someone to phone paramedic
13.	yell	m.	to assist if necessary
		n.	unconsciousness
		o.	victim

****

The following facts are now true:
compression breathing cycles are performed, paramedic is phoned,
victim not breathing is observed, airway is open, victim is on_back,
and victim is unconscious.
What operator do you choose? [9,k]. OK.
#################### (Next Screen Display) ####################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	breathing
3.	instruct	c.	breaths
4.	locate	d.	compression position
5.	open	e.	compressions
6.	perform	f.	emergency breathing
7.	phone	g.	finger sweep
8.	position	h.	head
9.	recheck	i.	"help"
10.	retilt	j.	paramedic
11.	stop	k.	pulse
12.	standby	l.	someone to phone paramedic
13.	yell	m.	to assist if necessary
		n.	unconsciousness
		o.	victim

****

The following facts are now true:
still_no_pulse is true, paramedic is phoned, victim not breathing is observed,
airway is open, victim is on_back, and victim is unconscious.
What operator do you choose? [1,n].
That will not affect anything.

##################### (Next Screen Display) ####################
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	breathing
2.	give	b.	breaths
3.	instruct	c.	compression breathing cycle(s)
4.	locate	d.	compression position
5.	open	e.	compressions
6.	perform	f.	cpr
7.	phone	g.	emergency breathing
8.	position	h.	finger sweep
9.	recheck	i.	head
10.	retilt	j.	"help"
11.	stop	k.	paramedic
12.	standby	l.	pulse
13.	yell	m.	to assist if necessary
		n.	unconsciousness
		o.	victim

****

The following facts are now true:
still_no_pulse is true, paramedic is phoned, victim not breathing is observed, airway is open, victim is on_back, and victim is unconscious.
What operator do you choose? [2,b].

##################### (Next Screen Display) ####################

Your selection: give breaths

Enter the NUMBER of times the operator should be performed.

- O R -

Select from the following by entering the appropriate letter:

a. temporarily
b. forever
c. until no longer necessary, rechecking pulse every few minutes
d. 2 to 3
e. 6 to 10
f. 7 to 15

ENTER YOUR CHOICE: 2.
OK.

########## (Next Screen Display) ##########
POSSIBLE CHOICES FOR OPERATOR SELECTION ARE:

1.	check	a.	abdominal thrusts
2.	give	b.	airway
3.	instruct	c.	bleeding
4.	locate	d.	breathing
5.	open	e.	breaths
6.	perform	f.	compression breathing cycle(s)
7.	phone	g.	compressions
8.	position	h.	cpr
9.	recheck	i.	emergency breathing
10.	retilt	j.	head
11.	stop	k.	"help"
12.	standby	l.	paramedic
13.	yell	m.	pulse
		n.	to assist if necessary
		o.	unconsciousness

****
The following facts are now true:
sixth pair of breaths are given, still_no_pulse is true, paramedic is phoned,
victim not breathing is observed, airway is open, victim is on_back, and victim is unconscious.
What operator do you choose? [6,f].

########## (Next Screen Display) ##########

Your selection: *perform compression* breathing cycle(s)

Enter the NUMBER of times the operator should be performed.

- O R -

Select from the following by entering the appropriate letter:
a. temporarily
b. forever
c. until no longer necessary, rechecking pulse every few minutes
d. 2 to 3
e. 6 to 10
f. 7 to 15

ENTER YOUR CHOICE: c.
OK.

########## (Next Screen Display) ##########
Full list of your actions: [check_if_unconscious,yell_for_help,position_victim,
open_airway,check_if_breathing,stop_bleeding,standby_to_assist,
give_two_breaths, check_pulse,instruct_someone_to_phone_paramedic,
locate_compression_position, perform_fifteen_compressions,give_two_breaths,
perform_three_compression_breathing_cycles, recheck_pulse,give_two_breaths,
perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes]

The operators you had difficulty in applying were:
yell_for_help

The operators you incorrectly applied were:
perform_fifteen_compressions

The operators you had difficulty in identifying correct repetitions:
give_two_breaths

The operators for which you requested help were:
open_airway

Your total time for responses was 9 minutes, 34 seconds.
This equates to an average response time of 22 seconds per question.

ENTER any letter key to return to main menu: c.

########## (END DEMONSTRATION) ##########

APPENDIX D - SOURCE CODE

```
{ file: cpr }

/*************************************************************************/
/*********************** DIRECTIONS ****************************/
/*************************************************************************/

  1. Load prolog with increased parameters:
     /usr/local/prolog -a 500 -g 500 -h 500
  2. Load the file "CPR" by typing [cpr].

  The following files are required:
     metutor
     mecpr
     utilities
     actions
     quit
     help
     datetemp

  CPR will take control and automatically begin execution of the tutor

/*************************************************************************/
/********************** PROGRAM DRIVER ***************************/
/*************************************************************************/

go :- read(EOF), consult(utilities), clear, intro, abolish(intro,0), clear,
     help_prompt, display_menu, !, choice(Choice), clear, initiate_action(Choice), !.

initiate_action(1) :-  consult(help), clear, help, abolish(choice,1),
     display_menu, choice(Choice), clear, initiate_action(Choice).
initiate_action(4) :- consult(quit), clear, execute(quit).
initiate_action(2) :- consult(actions), clear, execute(actions), clear,
     abolish(choice,1), display_menu, choice(Choice), clear, initiate_action(Choice).
initiate_action(Cat) :- initialization_performed, sourcefile(Cat, Filename),
     clear, user_experience, standby3, clear, execute(Filename),
     abolish(choice,1), display_menu, choice(Choice), initiate_action(Choice), !.
initiate_action(Cat) :- initialize_screen, files(Files),
     load_files(Files), asserta(initialization_performed), initiate_action(Cat).
```

```
/*************************************************************************/
/******************** INITIALIZATIONS ********************************/
/*************************************************************************/

session(0).
time_index(0).
questions_ans(0).

/*************************************************************************/
/********************** SOURCEFILES ********************************/
/*************************************************************************/

files([metutor, mecpr, actions]).

sourcefile(2,actions).          /* CPR steps are listed for reference */
sourcefile(3,mecpr).            /* CPR skill test using means_ends  */

/*************************************************************************/
/********************** MENUS & MESSAGES ********************************/
/*************************************************************************/

intro :- clear, intro_screen, standby7, clear, user_id, clear,
     abolish(user_id,1), abolish(intro_screen,0).

intro_screen :- clear, blanklines(7), tab(25),
     write('CCCCCC  PPPPPP  RRRRRR'), nl, tab(25),
     write('CC      PP  PP  RR  RR'), nl, tab(25),
     write('CC      PPPPPP  RRRRRR'), nl, tab(25),
     write('CC      PP      RR RR '), nl, tab(25),
     write('CCCCCC  PP      RR  RR'), nl, blanklines(6),
     write('  PROLOG REMINDER:
     ENTRIES MUST BE FOLLOWED BY A PERIOD (".")'),
     nl, standby3.

user_id :- blanklines(5), tab(20), write('Welcome to the CPR tutor'), nl,
     blanklines(10), write('Please enter your name: '), read(Name),
     asserta(user(Name)), nl.

user_experience :- clear, checkretract(user_level(L)), blanklines(10), tab(25),
     write('Level of experience'), nl, tab(25), write('-----------------'),
     nl, tab(25), write('    1. Novice'), nl, tab(25), write('    2. Intermediate'),
     nl, tab(25), write('    3. Advanced'), blanklines(3), write(' Enter 1., 2., or 3. : '),
     read(Selection), validate(Selection,3,Expertise), assert_level(Expertise), blanklines(2),
     tab(20), write('    Good luck '), user(Name), write(Name), write('!!!'), nl.
```

```
assert_level(1) :- asserta(user_level(novice)).
assert_level(2) :- asserta(user_level(intermediate)).
assert_level(3) :- asserta(user_level(advanced)).

initialize_screen :- blanklines(5), tab(33), write('C P R  Tutor'),
    blanklines(3), tab(30), write('INITIALIZING FILES'), blanklines(3),
    tab(32), write('Please standby'), blanklines(7).

help_prompt :- first_time_user, blanklines(2), tab(2), user(Name), write(Name),
    write(', if you are a first time user, suggest you first request HELP'),
    nl,retract(first_time_user).
help_prompt.
first_time_user.

display_menu :- blanklines(4), tab(10), writechar(*,53), nl, tab(10), write('*'), tab(51),
    write('*'), nl, tab(10), write('*     1. HELP                                  *'),nl,
    tab(10), write('*     2. Review CPR procedures                 *'),
    nl, tab(10),
    write('*     3. CPR skill test                          *'),nl, tab(10),
    write('*     4. QUIT                                  *'), nl, tab(10),
    write('*'), tab(51), write('*'), nl, tab(10), write('*'), tab(51), write('*'), nl,
    tab(10), write('*    ENTER YOUR CHOICE AS 1., 2., 3., or 4.        *'),nl,
    tab(10), write('*'), tab(51), write('*'), nl, tab(10), writechar(*,53),
    nl, blanklines(3), checkretract(choice(Choice)), !, read(Choice),
    validate(Choice,4,Selection), !, asserta(choice(Choice)).

do_intro :- intro(T), blanklines(5), writechar(*,79), nl, write(T), nl,
    writechar(*,79), blanklines(5), !.
do_intro.

menu(O,novice,NOL) :- op_list(OL), reduce_list(OL,O,5,NOL), blanklines(3),
    writechar(#,79), blanklines(3), nl, write('POSSIBLE OPERATORS ARE: '),
    nl, nl, writemenu(NOL), nl, nl.
menu(O,intermediate,NOL) :- op_list(OL), reduce_list(OL,O,15,NOL),
    writechar(#,79), nl, write('POSSIBLE OPERATORS ARE: '), nl,
    writemenu(NOL).
menu(O,advanced,NAIL) :- variable(Varlist), writechar(#,79), nl,
    write('POSSIBLE CHOICES FOR OPERATOR SELECTION ARE: '), nl, verb(VL),
    lengths(L), action_id(AIL), get_op(O2,O), reduce_list(AIL,O2,15,NAIL),
    printmenu(VL,L,NAIL,Varlist,1), directions.
```

```
get_op(O2,O) :- opcode([_,O2],O).
get_op(O2,O) :- opcode([_,O2],Q,O).

writemenu([]) :- !.
writemenu(OL) :- length(OL,N), writemenu2(OL,N), asserta(num_ops(N)).
writemenu2([I|ROL],N) :- length(ROL,N2), N3 is N - N2, tab(3), spacer(N3),
    write(N3), write('. '), write(I), nl, writemenu2(ROL,N).
writemenu2([],N) :- !.

printmenu([],[],[],Varlist,N) :- !.
printmenu([],[],[A1|AROL],Varlist,N) :- tab(34), nth_element(Varlist,N,Var),
    N1 is N + 1, write(Var), write('. '), write(A1), nl,
    printmenu([],L,AROL,Varlist,N1).
printmenu([V1|VROL],[L|ROL],[],Varlist,N) :- tab(5), spacer(N), write(N),
    write('. '), write(V1), nl, N1 is N + 1,
    printmenu(VROL,ROL,[],Varlist,N1).
printmenu([V1|VROL],[L|ROL],[A1|AROL],Varlist,N) :- tab(5), spacer(N), write(N),
    write('. '), write(V1), S is 25 - L, tab(S), nth_element(Varlist,N,Var),
    N1 is N + 1, write(Var), write('. '), write(A1), nl,
    printmenu(VROL,ROL,AROL,Varlist,N1).

qual_menu([A1,A2],Sel) :- clear, nl,nl,nl, write('Your selection:  '),
    write(A1), write(' '), write(A2), blanklines(3), writechar(-,79),
    nl, nl, tab(5),
    write('Enter the NUMBER of times the operator should be performed.'),
    nl, nl, tab(30), write('- O R -'), nl, nl, tab(5),
    write('Select from the following by entering the appropriate letter:'),
    nl, quals(Quals), display_quals(Quals),nl,nl, writechar(-,79), nl, nl,
    write('ENTER YOUR CHOICE:  '), read(Choice),
    validate3(Choice,Quals,Sel).

directions :-  first_time, writechar(-,79), nl,
    write('ENTER YOUR CHOICE BY SELECTING ONE ITEM FROM
    EACH COLUMN'), nl, write('Example: To select "check pulse",
    enter "[2,k]." '), nl, retract(first_time).
directions.
first_time.
```

```
/*************************************************************************/
/************** SUMMARIZE USER SESSION ***************************/
/*************************************************************************/

summary(Oplist) :- clear, nl, write('Full list of your actions: '),
     write(Oplist), nl, nl, sum_errors(EL), sum_errors2(EL2),
     sum_qual_errors(QEL), sum_help(HL), time_index(T),
     write('Your total time for responses was '), M is T/60, M1 is floor(M),
     write(M1), write(' minutes, '), S is T mod 60,
     write(S), write(' seconds.'), questions_ans(QA), Ave is T/QA, nl,
     write('This equates to an average response time of '), A is floor(Ave),
     write(A),write(' seconds per question.'), nl,
     store_session(Oplist,T,QA,EL,EL2,QEL,HL).

store_session(Oplist,T,QA,EL,EL2,QEL,HL) :- user(Name), session(N),
     asserta(session_summary(Name,N,Oplist,T,QA,EL,EL2,QEL,HL)).

reset_counters :- retract(time_index(C)), retract(questions_ans(QA)),
     asserta(time_index(0)), asserta(questions_ans(0)), abolish(error,4),
     abolish(qual_error,1), abolish(help,4).

sum_errors(EL) :- bagof(O,error(_,O,_,_),EL), !, length(EL,N), N > 0, !,
     write("The operators you had difficulty in applying were: '), nl,
     writelistalt(EL), nl, nl.
sum_errors([]).

sum_errors2(EL2) :- bagof(O2,error(O2,_,_,_),EL2), !, length(EL2,N), N>0, !,
     write('The operators you incorrectly applied were: '), nl,
     writelistalt(EL2), nl, nl.
sum_errors2([]).

sum_qual_errors(QEL) :- bagof(O,qual_error(O),QEL),
     write('The operators you had difficulty in identifying correct repetitions:'),
     nl, writelistalt(QEL), nl, nl.
sum_qual_errors([]).

sum_help(HL) :- bagof(O,help(_,O,_,_),HL), !, length(HL,N), N>0, !,
     write('The operators for which you requested help were: '), nl,
     writelistalt(HL), nl, nl.
sum_help([]).

:- go.
```

```
{ file: mecpr }

intro('     This is a test of your CPR skills - skills that could save a life!').

/*********************************************************************/
/******************  EXPERT MODULE  ******************************/
/*********************************************************************/

opcode([check,breathing], check_if_breathing).
opcode([check,breaths], check_if_breathing).
recommended([observed('victim not breathing')], check_if_breathing).
precondition(check_if_breathing,
    [on_back(victim), open(airway), not(breathing(victim)), not(stopped_breathing)]).
deletepostcondition(check_if_breathing, []).
addpostcondition(check_if_breathing, [observed('victim not breathing')]).
randsubst(check_if_breathing, [[observed('victim not breathing'), breathing(victim), 0.3]]).

opcode([check,unconsciousness], check_if_unconscious).
opcode([check,victim], check_if_unconscious).
recommended([unconscious(victim)], check_if_unconscious).
precondition(check_if_unconscious, []).
deletepostcondition(check_if_unconscious, [person_collapsed]).
addpostcondition(check_if_unconscious, [unconscious(victim)]).
randsubst(check_if_unconscious, [[none, bleeding(victim), 0.5]]).

opcode([check,pulse], check_pulse).
recommended([observed('no pulse')], check_pulse).
recommended([not(performed('finger sweep'))], check_pulse).
precondition(check_pulse, [unconscious(victim), observed('victim not breathing'),
    not(observed(pulse)), given('first pair of breaths'), not(obstructed(airway))]).
deletepostcondition(check_pulse, [performed('finger sweep')]).
addpostcondition(check_pulse, [observed('no pulse')]).
randsubst(check_pulse, [[observed('no pulse'), observed(pulse), 0.2]]).

opcode([recheck,pulse], recheck_pulse).
recommended([still_no_pulse], recheck_pulse).
recommended([not(performed('compression breathing cycles'))], recheck_pulse).
precondition(recheck_pulse,
    [unconscious(victim), performed('compression breathing cycles'),
    not(given('first pair of breaths')), not(still_no_pulse)]).
deletepostcondition(recheck_pulse, [phoned(paramedic)],
    [performed('compression breathing cycles')]).
deletepostcondition(recheck_pulse, []).
addpostcondition(recheck_pulse, [still_no_pulse]).
randsubst(recheck_pulse, [[still_no_pulse, observed(pulse), 0.2]]).
```

```
precondition(give_two_breaths,
    [unconscious(victim), observed('victim not breathing'),
    not(located('compression position')), not(breathing(victim)),
    not(performed('compression breathing cycles')),
    not(observed('no pulse')), not(obstructed(airway))]).

opcode([instruct,'someone to phone paramedic'],
    instruct_someone_to_phone_paramedic).
recommended([phoned(paramedic)], instruct_someone_to_phone_paramedic).
recommended([not(available(assistance))], instruct_someone_to_phone_paramedic).
precondition(instruct_someone_to_phone_paramedic,
    [available(assistance), observed('no pulse')]).
deletepostcondition(instruct_someone_to_phone_paramedic,
    [available(assistance)]).
addpostcondition(instruct_someone_to_phone_paramedic, [phoned(paramedic)]).

opcode([locate,'compression position'], locate_compression_position).
recommended([located('compression position')], locate_compression_position).
recommended([not(given('first pair of breaths'))],
    locate_compression_position).
precondition(locate_compression_position,
    [observed('no pulse'), observed('victim not breathing'),
    not(available(assistance))]).
deletepostcondition(locate_compression_position,
    [given('first pair of breaths')]).
addpostcondition(locate_compression_position,[located('compression position')]).

opcode([open,airway], open_airway).
recommended([open(airway)], open_airway).
precondition(open_airway, [on_back(victim), unconscious(victim)]).
deletepostcondition(open_airway, []).
addpostcondition(open_airway, [open(airway)]).
randsubst(open_airway, [[unavailable(assistance), available(assistance), 0.3,
    'Someone has responded to your yell for help.']]).

opcode([perform,'abdominal thrusts'], '6 to 10', perform_abdominal_thrusts).
opcode([give,'abdominal thrusts'], '6 to 10', perform_abdominal_thrusts).
recommended([performed('abdominal thrusts')], perform_abdominal_thrusts).
precondition(perform_abdominal_thrusts,
    [retilt(head), given('first pair of breaths')]).
deletepostcondition(perform_abdominal_thrusts, []).
addpostcondition(perform_abdominal_thrusts,
    [performed('abdominal thrusts'), obstructed(airway)]).
```

```
opcode([perform,'finger sweep'], perform_finger_sweep).
opcode([give,'finger sweep'], perform_finger_sweep).
recommended([performed('finger sweep')], perform_finger_sweep).
recommended([not(obstructed(airway))], perform_finger_sweep).
recommended([not(given('first pair of breaths'))], perform_finger_sweep).
recommended([not(performed('abdominal thrusts'))], perform_finger_sweep).
recommended([not(retilt(head))], perform_finger_sweep).
precondition(perform_finger_sweep, [performed('abdominal thrusts')]).
deletepostcondition(perform_finger_sweep,
    [retilt(head), performed('abdominal thrusts'), obstructed(airway),
    given('first pair of breaths')]).
addpostcondition(perform_finger_sweep, [performed('finger sweep')]).

opcode([perform,'compression breathing cycle(s)'],
    'until no longer necessary, rechecking pulse every few minutes',
    perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes).
opcode([give,'compression breathing cycle(s)'],
    'until no longer necessary, rechecking pulse every few minutes',
    perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes).
recommended([on_scene('victim is breathing and_or paramedic')],
    perform_compression_breathing_cycles_rechecking_pulse_every_few_minutes).
precondition(perform_compression_breathing_cycles_rechecking_pulse_every_few_
    minutes, [still_no_pulse, phoned(paramedic), given('sixth pair of breaths')]).
deletepostcondition(perform_compression_breathing_cycles_rechecking_pulse_every_
    few_minutes, [given('sixth pair of breaths'), phoned(paramedic), on_back(victim),
    open(airway), observed('victim not breathing'), still_no_pulse, unconscious(victim)]).
addpostcondition(perform_compression_breathing_cycles_rechecking_pulse_every_
    few_minutes, [on_scene('victim is breathing and_or paramedic')]).

opcode([perform,'emergency breathing'], perform_emergency_breathing).
opcode([give,'emergency breathing'], perform_emergency_breathing).
recommended([still_no_pulse], perform_emergency_breathing).
recommended([observed('no pulse')], perform_emergency_breathing).
recommended([not(observed(pulse))], perform_emergency_breathing).
precondition(perform_emergency_breathing,
    [unconscious(victim), observed('victim not breathing'), observed(pulse)]).
deletepostcondition(perform_emergency_breathing, [observed(pulse)]).
addpostcondition(perform_emergency_breathing,
    [given('first pair of breaths')], [observed('no pulse')]).
addpostcondition(perform_emergency_breathing,
    [not(given('first pair of breaths'))], [still_no_pulse]).
addpostcondition(perform_emergency_breathing, [error-in-perform-emergency-breathing]).
randsubst(perform_emergency_breathing, [[unconscious(victim), unconscious(victim), 1.0,
    'The heart has stopped beating. What should you do now?']]).
```

```
opcode([perform,compressions],15,perform_fifteen_compressions).
opcode([give,compressions],15,perform_fifteen_compressions).
recommended([performed('fifteen compressions')], perform_fifteen_compressions).
recommended([not(located('compression position'))],
    perform_fifteen_compressions).
recommended([not(observed('no pulse'))]).
precondition(perform_fifteen_compressions,
    [located('compression position'), observed('no pulse')]).
deletepostcondition(perform_fifteen_compressions,
    [located('compression position'), observed('no pulse')]).
addpostcondition(perform_fifteen_compressions,
    [performed('fifteen compressions')]).

opcode([perform,'compression breathing cycle(s)'],3,
    perform_three_compression_breathing_cycles).
opcode([give,'compression breathing cycle(s)'],3,
    perform_three_compression_breathing_cycles).
recommended([performed('compression breathing cycles')],
    perform_three_compression_breathing_cycles).
recommended([not(performed('fifteen compressions'))],
    perform_three_compression_breathing_cycles).
recommended([not(given('second pair of breaths'))],
    perform_three_compression_breathing_cycles).
precondition(perform_three_compression_breathing_cycles,
    [performed('fifteen compressions'), given('second pair of breaths')]).
deletepostcondition(perform_three_compression_breathing_cycles,
    [performed('fifteen compressions'), given('second pair of breaths')]).
addpostcondition(perform_three_compression_breathing_cycles,
    [performed('compression breathing cycles')]).

opcode([phone,paramedic], phone_paramedic).
recommended([phoned(paramedic)],phone_paramedic).
recommended([performed('compression breathing cycles')], phone_paramedic).
precondition(phone_paramedic,
    [still_no_pulse, unavailable(assistance), temporarily_stopped(cpr)]).
deletepostcondition(phone_paramedic,
    [unavailable(assistance), temporarily_stopped(cpr),
    performed('compression breathing cycles')]).
addpostcondition(phone_paramedic, [phoned(paramedic)]).
```

```
opcode([position,victim], position_victim).
recommended([on_back(victim)], position_victim).
precondition(position_victim, [unconscious(victim), yelled('"help"')]).
deletepostcondition(position_victim, [yelled('"help"')]).
addpostcondition(position_victim, [on_back(victim)]).
randsubst(position_victim, [[unavailable(assistance), available(assistance),
    0.3, 'Someone has responded to your yell for help.']]).

opcode([retilt,head], retilt_head).
recommended([retilt(head)], retilt_head).
recommended([not(obstructed(airway))], retilt_head).
precondition(retilt_head, []).
deletepostcondition(retilt_head,
    [obstructed(airway), given('first pair of breaths')]).
addpostcondition(retilt_head, [retilt(head)]).

opcode([standby,'to assist if necessary'], standby_to_assist).
recommended([not(breathing(victim))], standby_to_assist).
recommended([observed('victim not breathing')], standby_to_assist).
recommended([not(stopped(bleeding))], standby_to_assist).
precondition(standby_to_assist, [breathing(victim), not(bleeding(victim))]).
deletepostcondition(standby_to_assist, [breathing(victim), stopped(bleeding)]).
addpostcondition(standby_to_assist, [observed('victim not breathing'),
    stopped_breathing]).
randsubst(standby_to_assist, [[observed('victim not breathing'),
    observed('victim not breathing'), 1.0,
    'Breathing has stopped. What should you do now?']]).

opcode([stop,bleeding], stop_bleeding).
opcode([check,bleeding], stop_bleeding).
recommended([not(bleeding(victim))], stop_bleeding).
recommended([stopped(bleeding)], stop_bleeding).
precondition(stop_bleeding, [bleeding(victim)]).
deletepostcondition(stop_bleeding, [breathing(victim)], [bleeding(victim)]).
deletepostcondition(stop_bleeding, []).
addpostcondition(stop_bleeding, [breathing(victim)], [stopped(bleeding)]).
addpostcondition(stop_bleeding, []).
randsubst(stop_bleeding, [[breathing(victim), observed('victim not breathing'),
    0.4, 'Bleeding is stopped. However, the victim has stopped breathing.']]).
```

```
opcode([stop,cpr], temporarily, temporarily_cease_cpr).
opcode([stop,'compression breathing cycle(s)'], temporarily, temporarily_stop_cpr).
recommended([temporarily_stopped(cpr)], temporarily_stop_cpr).
precondition(temporarily_stop_cpr,
    [still_no_pulse, unavailable(assistance), not(given('second pair of breaths'))]).
deletepostcondition(temporarily_stop_cpr, []).
addpostcondition(temporarily_stop_cpr, [temporarily_stopped(cpr)]).

opcode([yell,'"help"'], yell_for_help).
recommended([yelled('"help"')], yell_for_help).
recommended([unavailable(assistance)], yell_for_help).
precondition(yell_for_help, []).
deletepostcondition(yell_for_help, [not(available(assistance))]).
addpostcondition(yell_for_help, [unavailable(assistance), yelled('"help"')]).
randsubst(yell_for_help,[[unavailable(assistance), available(assistance), 0.3,
    'Someone has responded to your yell for help.']]).

opcode([give,breaths],2,give_two_breaths).
opcode([perform,breaths],2,give_two_breaths).
recommended([given('sixth pair of breaths')], give_two_breaths).
recommended([given('second pair of breaths')], give_two_breaths).
recommended([given('first pair of breaths')], give_two_breaths).
recommended([not(stopped_breathing)], give_two_breaths).
/* precondition(give_two_breaths - for alphabetical order listed above */
deletepostcondition(give_two_breaths,
    [stopped_breathing, performed('finger sweep')]).
addpostcondition(give_two_breaths,
    [phoned(paramedic), still_no_pulse],
    [given('sixth pair of breaths')]).
addpostcondition(give_two_breaths,
    [on_back(victim), open(airway), performed('fifteen compressions')],
    [given('second pair of breaths')]).
addpostcondition(give_two_breaths,
    [open(airway), on_back(victim), not(phoned(paramedic))],
    [given('first pair of breaths')]).
addpostcondition(give_two_breaths, [error-in-give-two-breaths]).
randsubst(give_two_breaths,
    [[given('first pair of breaths'), obstructed(airway), 0.3,
    'Airway is obstructed - you are unable to breathe air into the victim.']]).
```

```
mod_randsubst(give_two_breaths, Postlist) :-
    member(obstructed(airway),Postlist),
    checkretract(randsubst(give_two_breaths,
    [[given('first pair of breaths'),_,_,_]])).

mod_randsubst(Op,Postlist).

nopref(check_if_unconscious, yell_for_help).

verb([check,give,instruct,locate,open,perform,phone,position,recheck,retilt,
    stop,standby,yell]).
lengths([5,4,8,6,4,7,5,8,7,6,4,7,4]).

action_id(['abdominal thrusts', airway, bleeding, breathing, breaths,
    'compression breathing cycle(s)', 'compression position', compressions,
    cpr, 'emergency breathing', 'finger sweep', head, '"help"', paramedic,
    pulse, 'someone to phone paramedic', 'to assist if necessary',
    unconsciousness, victim]).

quals([a,b,c,d,e,f]).
qualifier(a,temporarily).
qualifier(b,forever).
qualifier(c,'until no longer necessary, rechecking pulse every few minutes').
qualifier(d,'2 to 3').
qualifier(e,'6 to 10').
qualifier(f,'7 to 15').


execute(mecpr) :- tutor([person_collapsed],
    [on_scene('victim is breathing and_or paramedic')]), !, clear.
```

```
{ partial file: utilities }

/*************************************************************************/
/********************* VALIDATION ROUTINES ***************************/
/*************************************************************************/

validate(C,Max,C) :- integer(C), C>=1, C=<Max, !.
validate(C,Max,New) :- nl, nl, user(Name), write(Name), write(', '),
    write('your input is not within range of 1 to '), write(Max),
    write('.  '), write('Please reenter: '), read(Choice),
    validate(Choice,Max,New), !.

validate2([VO2,AO2],VL,AL,[VO2,AO2]) :- integer(VO2), VO2>=1, length(VL,Len),
    VO2=<Len, variable(Var), member(AO2,Var).
validate2(O2in,VL,AL,New) :-
    write('Input must be in the form [number,letter] and within specified
    range'), nl, write('Please reenter:  '), read(Newchoice),
    validate2(Newchoice,VL,AL,New), !.

validate3(C,Q,C) :- integer(C).
validate3(C,Q,S) :- member(C,Q), qualifier(C,S).
validate3(C,Q,New) :- write('Input must be either an integer or valid letter.'),
    nl, write('PLEASE REENTER:  '), read(C2), validate3(C2,Q,New).


/*************************************************************************/
/********************* UTILITY PREDICATES ***************************/
/*************************************************************************/

main_menu_return :-  blanklines(2), writechar(-,79), nl, nl,
    write('To return to main menu, enter "c." '), read(C).

load_files([]).
load_files([X|Y]) :- consult(X), load_files(Y).

blanklines(0) :- !.
blanklines(N) :- nl, N1 is N - 1, blanklines(N1).

/* clears screen */
clear :- system("clear").
clear.

/* system pauses for designated amount of time */
standby3 :- system("sleep 3").
standby3.
```

```
standby7 :- system("sleep 7").
standby7.

/*  repeatedly write the specified character N times  */
writechar(C,0) :- !.
writechar(C,N) :- write(C), N1 is N - 1, writechar(C,N1).

/* output the Nth element of the list */
nth_element([L|ROL],1,L).
nth_element([L|ROL],N,X) :- N1 is N - 1, nth_element(ROL,N1,X).

/* convert variable to number relating the position in list */
convert_var(Var,VL,N) :- length(VL,L), convert2(Var,VL,L,N).
convert2(Var,[Var|RVL],L,N) :- length(RVL,L2), N is L - L2.
convert2(Var,[Var1|RVL],L,N) :- convert2(Var,RVL,L,N).

/*  reduce a list into a list of N length - ensure operator O is NOT deleted */
reduce_list(OL,O,N,OL) :- length(OL,Len), Len =< N.
reduce_list(OL,O,N,NOL) :- get_time(X), length(OL,Len), T is (X mod Len) + 1,
     nth_element(OL,T,Delitem),
     ((Delitem = O, reduce_list(OL,O,N,NOL)) ;
     (delete(Delitem,OL,NL), reduce_list(NL,O,N,NOL))).

/* give a blank space before numbers under 10 */
spacer(N) :- N<10, write(' ').
spacer(N).

/*  write a list of items  - alternate way to that in natural language output */
writelistalt([]) :- nl, !.
writelistalt([H|T]) :- write(H), tab(1), writelistalt(T), !.

display_quals([]) :- !.
display_quals([H|T]) :- tab(10), write(H), write('. '), qualifier(H,Q),
     write(Q), nl, display_quals(T).


quitword(quit).

variable([a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z]).
```

```
{ file:actions }


execute(actions) :-
     clear, nl, tab(15), write('CPR OVERVIEW - MAJOR STEPS'), nl, nl,
     steps(Major_steps_ids),
     display_steps(Major_steps_ids), standby3, main_menu_return, clear.

display_steps([]):- !.
display_steps([H|T]) :- tab(2), spacer(H), write(H), write('. '),
     major_action([H],Desc,Substeps), writelistalt(Desc), display_steps(T).

steps([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]).

major_action([1],['Check if unconscious.'],2).
     sub_action([1,1],['Tap or gently shake victim.']).
     sub_action([1,2],['Shout "Are you ok?"']).

major_action([2],['Yell for help.'],0).

major_action([3],['Position victim.'],5).
     sub_action([3,1],['Roll victim onto back.']).
     sub_action([3,2],['Kneel facing victim, between hips and shoulders.']).
     sub_action([3,3],['Lean over victim, place hands on shoulder and hip.']).
     sub_action([3,4],['Roll victim toward you as single unit; move hand
        from shoulder to suppost back of head and neck.']).
     sub_action([3,5],['Place arm of victim nearest you alongside their body.']).

major_action([4],['Open airway.'],3).
     sub_action([4,1],['Place one hand on forehead of victim.']).
     sub_action([4,2],['Place fingers of other hand under bony part of
       lower jaw near chin.']).
     sub_action([4,3],['Tilt head and lift jaw - avoid closing mouth.']).

major_action([5],['Check if breathing.'],3).
     sub_action([5,1],['Maintain open airway.']).
     sub_action([5,2],['Place your ear over mouth and nose of victim.']).
     sub_action([5,3],['Look at chest, listen and feel for breathing for 3 to 5 seconds.']).

major_action([6],['Give two full breaths.'],4).
     sub_action([6,1],['Open airway.']).
     sub_action([6,2],['Pinch nose shut.']).
     sub_action([6,3],['Open your mouth wide, take deep breath, and make
       tight seal around outside of mouth of victim.']).
     sub_action([6,4],['Observe chest rise and fall, listen and feel for escaping air.']).
```

```
major_action([7],['Check pulse.'],4).
    sub_action([7,1],['Maintain head tilt with one hand on forehead.']).
    sub_action([7,2],['Locate Adam apple with middle and index fingers.']).
    sub_action([7,3],['Slide fingers down into groove of neck.']).
    sub_action([7,4],['Feel for cartoid pulse for 5 to 10 seconds.']).

major_action([8],['Instruct someone to phone paramedic - if assistance
      is available.'],0).

major_action([9],['Locate compression position.'],8).
    sub_action([9,1],['Kneel facing chest of victim.']).
    sub_action([9,2],['With middle and index fingers of hand nearest legs
      of victim, locate lower edge of rib cage on side closest to you.']).
    sub_action([9,3],['Follow rib cage to notch at lower end of
      breastbone.']).
    sub_action([9,4],['Place middle finger in notch and index finger on
      lower end of breastbone.']).
    sub_action([9,5],['Place heel of hand nearest head of victim on
      breastbone next to index finger of hand used to find notch.']).
    sub_action([9,6],['Place heel of hand used to locate notch directly on
      top of heel of other hand.']).
    sub_action([9,7],['Keep fingers off chest of victim.']).
    sub_action([9,8],['Position shoulders over hands with elbows locked
      and arms straight.']).

major_action([10],['Perform 15 compressions.'],2).
    sub_action([10,1],['Compress breastbone 1 1/2 to 2 inches at a rate
      of 80 to 100 compressions per minute.']).
    sub_action([10,2],['Compress down and up smoothly, keeping contact
      with chest at all times.']).

major_action([11],['Give two full breaths.'],1).
    sub_action([11,1],['See step 6']).

major_action([12],['Perform 3 compression/breathing cycles.'],1).
    sub_action([12,1],['Do cycles of 15 compressions and 2 breaths.']).

major_action([13],['Recheck pulse.'],3).
    sub_action([13,1],['Tilt head.']).
    sub_action([13,2],['Locate cartoid pulse and feel for 5 seconds.']).
    sub_action([13,3],['For more details, see step 7']).
```

```
major_action([14],['Temporarily cease cpr to phone paramedic - if phonecall
      not already made.'],0).

major_action([15],['Give two full breaths.'],1).
    sub_action([15,1],['See step 6.']).

major_action([16],['Perform compression_breathing cycles until no longer
    applicable.'],3).
    sub_action([16,1],['Locate correct hand position.']).
    sub_action([16,2],['Continue cycles of 15 compressions and 2 breaths.']).
    sub_action([16,3],['Recheck pulse every few seconds.']).




- - - - - - - - - - - - - - - - -

{ file: quit }

execute(quit) :-
    clear, write('This is a stub of the quit file - data is in raw output form.'),
    nl, write('It summarizes the action of the user during the session.'), nl,
    writechar(*,79), nl, session_summary(Name,N,Oplist,T,QA,EL,EL2,QEL,HL),
    !, listing(session_summary), clear, halt.
execute(quit) :- clear, halt.

execute(quit2) :-
    clear, blanklines(10), tab(20), write('SESSION TERMINATED AT USER REQUEST'),
    blanklines(10), halt.
```

```
{ file: help }


help :- clear, nl, tab(35), write('C P R'), nl, nl, nl,tab(5),
        write('One out of every two people in the United States can expect to '), nl,
        tab(5), write('die from a heart attack or a related disease or failure.'), nl, tab(5),
        write('The American Red Cross offers adult CPR courses in an effort to '),
        nl, tab(5), write('reduce this drastic death toll.'), nl, nl, tab(5),
        write('This CPR Tutor is designed to help maintain your level of '), nl, tab(5),
        write('proficiency in knowing how to provide CPR treatment at the scene '), nl, tab(5),
        write('of an emergency. Based upon your level of experience, you will '), nl, tab(5),
        write('be prompted by the tutor for your response when confronted with '),
        nl, tab(5), write('the current state of the emergency.'), blanklines(8),
        write('ENTER any letter key to continue: '), read(L), clear, nl,
        tab(32), write('NOVICE'), nl, tab(32), writechar(-,6), nl, nl, tab(5),
        write('As a novice user you will be given a list of 5 possible operators, '),
        nl, tab(5), write('one of which is the correct response. '), nl, nl,
        nl, tab(30), write('INTERMEDIATE'), nl, tab(30), writechar(-,12), nl, nl,
        tab(5), write('As an intermediate user you will be given a list of 15 possible '),
        nl, tab(5), write('operators, one of which is the correct response.'), nl, nl, nl,
        tab(32), write('ADVANCED'), nl, tab(32), writechar(-,8), nl, nl, tab(5),
        write('As an advanced user you will be presented with two columns.'), nl, tab(5),
        write('Your response will be a selection of one number from column one and '),
        nl, tab(5), write('one letter from column two in the format "[#,letter]" such that'),
        nl, tab(5), write('a phase representing the opertor will be formed.'),
        nl, nl, write('ENTER any letter to continue: '), read(M), clear, nl,
        tab(35), write('HELP'), nl, tab(35), writechar(-,4), nl, nl, tab(5),
        write('At any time you are unsure of the correct response and desire help '),
        nl, tab(5), write('you can enter "help." or simply "h.". This will prompt the '),
        nl, tab(5), write('system to give you the preferred response.'), nl, nl, nl,
        tab(35), write('QUIT'), nl, tab(35), writechar(-,4), nl, nl, tab(5),
        write('At the end of each session you will be provided a summary of your '),
        nl, tab(5), write('actions. If you desire to end a session prior to the logical '),
        nl, tab(5), write('completion, entering "quit." will abort the program and return '),
        nl, tab(5), write('you to the system prompt. No summary of the aborted session '),
        nl, tab(5), write('will be provided.'), main_menu_return, clear.
```

APPENDIX E - ROWE SOURCE CODE

NOTE: The code contained herein was written by, and used with the permission of Professor Neil C. Rowe of the Naval Postgraduate School.

Predicates modified by the author of this thesis are indicated by a preceeding asterisk.
Additional predicates inserted by the author of this thesis are preceeded by a plus sign.

```
{ file: metutor }

/*************************************************************************/
/************************* TUTOR MODULE **************************/
/*************************************************************************/

*tutor(State,Goal) :- not(check_obvious_errors), issue_warnings, randinit,
     clear, nl, do_intro, write('     Your objectives: '),
     writelist(Goal,state), write('.'), nl, uniqueassert(top_goal(Goal)),
     bagof(X,P^precondition(X,P),XL), uniqueassert(op_list(XL)), blanklines(4),
     write('     Wait a moment while I analyze the problem thoroughly.'),
     nl, once_means_ends(State,Goal,Oplist2,Goalstate2), !,
     uniqueassert(top_solution(Statelist)), abolish(mainline_states,4),
     means_ends_tutor(State,Goal,Oplist,Goalstate,[]), !,
     summary(Oplist), reset_counters, main_menu_return, nl, clear, !.
tutor(State,Goal) :-
     write('Too bad: a solution is now impossible.'), nl, !.

*means_ends_tutor(State,Goal,[],State,Stack) :-
     writetrace(State,Goal), difference(Goal,State,[]), !.
means_ends_tutor(State,Goal,Oplist,State,Stack) :-
     member([State,Goal],Stack), !, fail.
means_ends_tutor(State,Goal,Oplist,Goalstate,Stack) :-
     not(once_means_ends(State,Goal,Oplist,Goalstate)), !, fail.
means_ends_tutor(State,Goal,Oplist,Goalstate,Stack) :-
     difference(Goal,State,D), applicable_op(D,Op), precondition(Op,Prelist),
     all_achievable(State,Prelist), !,
     means_ends_tutor(State,Prelist,Preoplist,Prestate,[[State,Goal]|Stack]),
     !,met(State,Goal,Oplist,Goalstate,Stack,Prelist,Preoplist,Prestate,Op,D).
```

```
met(State,Goal,Preoplist,Prestate,Stack,Prelist,Preoplist,Prestate,Op,D) :-
     difference(Goal,Prestate,[]), !.
met(State,Goal,Oplist,Goalstate,Stack,Prelist,Preoplist,Prestate,Op,D) :-
     difference(Goal,Prestate,D2), not(applicable_op(D2,Op)), !,
     means_ends_tutor(Prestate,Goal,Oplist2,Goalstate,[]),
     append(Preoplist,Oplist2,Oplist).
*met(State,Goal,Oplist,Goalstate,Stack,Prelist,Preoplist,Prestate,Op,D) :-
     check_with_student(Op,Prestate,D,NewOp),
     get_deletepostcondition(NewOp,Prestate,Deletepostlist),
     deleteitems(Deletepostlist,Prestate,Prestate2),
     get_addpostcondition(NewOp,Prestate,Addpostlist),
     union(Addpostlist,Prestate2,Postlist2),
     do_randsubst(NewOp,Postlist2,Postlist), mod_randsubst(NewOp,Postlist),
     check_mainline_return(Postlist), !,
/* Note last arg. below empty to allow for randsubst returning to past state */
     means_ends_tutor(Postlist,Goal,Postoplist,Goalstate,[]),
     append(Preoplist,[NewOp|Postoplist],Oplist).

/*** Tutoring rules ***/

*check_with_student(O,S,D,NO) :- standby3, clear, user_level(Level),
     menu(O,Level,NL), writechar(*,79), nl,
     write('The following facts are now true:'), nl,
     writelist(S,state), write('.'), nl,
     write('What operator do you choose? '), get_time(T), read(O2in),
     get_time(T2), time_index(Counter), questions_ans(QA),
     retract(questions_ans(QA)), retract(time_index(Counter)),
     Elapsed_time is T2 - T, New_counter is Counter + Elapsed_time,
     asserta(time_index(New_counter)), QA1 is QA + 1,
     asserta(questions_ans(QA1)), user_level(Level),
     interpret_input(NL,Level,S,O2in,O2), handle_student_op(O2,O,S,D,NO).

handle_student_op(O,O,S,D,O) :- !, write('OK.'), nl.
*handle_student_op(O2,O,S,D,NO) :- helpword(O2), !, asserta(help(O2,O,S,D)),
     write('The operator you now need to perform is: '), nl, tab(5),
     write(O), nl, standby3, check_with_student(O,S,D,NO).
*handle_student_op(O2,O,S,D,NO) :- quitword(O2), clear, consult(quit),
     execute(quit2), !.
*handle_student_op(O2,O,S,D,NO) :- not(nopref(O2,O)), not(nopref(O,O2)),
     asserta(error(O2,O,S,D)), fail.
```

```
*handle_student_op(perform_fifteen_compressions,O,S,D,NO) :-
    not(member(observed('no pulse'),S)), !, clear, blanklines(10),
    writechar(@,79), nl, nl,
    write('It is important that you check the cartoid pulse for 5 to 10 '),
    nl, write('seconds before starting CPR.'), nl, nl,
    write('IT IS DANGEROUS TO PERFORM CHEST COMPRESSIONS IF THE
    HEART IS '), write('BEATING.'), nl, nl, writechar(@,79), nl, standby3,
    check_with_student(O,S,D,NO).
handle_student_op(O2,O,S,D,NO) :- precondition(O2,PO2), difference(PO2,S,D2),
    not(D2=[]), !, write('That operator requires that '),
    writelist(D2,precond), write('.'), nl, standby3,
    check_with_student(O,S,D,NO).
*handle_student_op(stop_bleeding,O,S,D,NO) :- apply_op(O2,S,S),
    write('YOU MUST be primarily concerned with the pulse and breath
    of the victim!!!'), nl,
    write('You must temporarily ignore any injuries.'), nl,
    check_with_student(O,S,D,NO).
handle_student_op(O2,O,S,D,NO) :- apply_op(O2,S,S),
    write('That will not affect anything.'), nl,
    check_with_student(O,S,D,NO).
handle_student_op(O2,O,S,D,NO) :- apply_op(O2,S,S2), top_goal(G),
    not(once_means_ends(S2,G,OL2,GS2)), !,
    write('You cannot ever succeed if you do that.'), nl,
    check_with_student(O,S,D,NO).
handle_student_op(O2,O,S,D,O2) :- top_goal(G), apply_op(O,S,S3),
    apply_op(O2,S,S2), compare_solutions(S3,G,OL3,GS3,S2,G,OL2,GS2),
    subsequence([O|OL3],OL2), !, apply_ops([O|OL3],S,SL,GS4),
    elimdups(SL,ESL), asserta(mainline_states(ESL,O2,S,O)),
    write('That does not seem immediately helpful, but I will try it.'), nl.
handle_student_op(O2,O,S,D,O2) :- (nopref(O2,O);nopref(O,O2)), !,
    write('OK.'), nl.
handle_student_op(O2,O,S,D,O2) :- top_goal(G),
    once_means_ends(S,G,OL,FS), not(member(O2,OL)), !,
    write('I will try it, but it is not recommended or needed for the
    problem.'), nl.
handle_student_op(O2,O,S,D,O2) :- top_goal(G), difference(G,S,D2),
    all_achievable(S,D2), applicable_op(D2,O3), precondition(O3,PL),
    least_common_op(S,G,O,O2,PL,Groot), !,
    write('I will try it, but it is not recommended first when '),
    difference(Groot,S,D5), delete_uncreatable(D5,D6),
    permutation(D6,D7), writelist(D7,precond), write('.'), nl.
handle_student_op(O2,O,S,D,O2) :-
    write('Not the operator I would choose, but let us try it.'), nl, !.
```

```
/* Intermediate predicates used by the tutor */

+interpret_input(NL,_,_,O2in,O2) :- helpword(O2in), helpword(O2).
+interpret_input(NL,_,_,O2in,O2) :- quitword(O2in), quitword(O2).
+interpret_input(NL,novice,S,O2in,O2) :- num_ops(N), validate(O2in,N,O3), !,
    nth_element(NL,O3,O2).
+interpret_input(NL,intermediate,S,O2in,O2) :- num_ops(N),
    validate(O2in,N,O3), !, nth_element(NL,O3,O2).
+interpret_input(NL,advanced,S,O2in,O2) :- verb(VL),
    validate2(O2in,VL,NL,[VO2,AO2]), check_opcode(O,VO2,AO2,S,VL,NL,O2).
+check_opcode(O,A1,A2,S,VL,AL,O2) :- nth_element(VL,A1,A1code),
    variable(Varlist), convert_var(A2,Varlist,N), nth_element(AL,N,A2code),
    get_opcode(O,[A1code,A2code],O2), !.
+check_opcode(O,A1,A2,S,VL,AL,O2) :-
    write('Your input is not an valid operator.'),
    write('  Please try again. '), nl, standby3, clear, menu(O,advanced,NL),
    writechar(*,79), nl, write('The following facts are true:'), nl,
    writelist(S,state), write('.'), nl,
    write('What operator do you choose?'),
    read(New), interpret_input(NL,advanced,S,New,O2).

+get_opcode(O,[A1,A2],O2) :- opcode([A1,A2],O2).
+get_opcode(O,[A1,A2],O2) :- opcode([A1,A2],_,_), qual_menu([A1,A2],Sel),
    verify_opcode(O,[A1,A2],Sel,O2), !, checkretract(missed_once).

+verify_opcode(O,[A1,A2],Sel,O2) :- opcode([A1,A2],Sel,O2).
+verify_opcode(O,[A1,A2],Sel,O2) :- not(missed_once), asserta(missed_once),
    write(Sel), write(' is not correct.'), nl, write('Try again:  '),
    read(New), verify_opcode(O,[A1,A2],New,O2).
+verify_opcode(O,[A1,A2],Sel,O2) :- write(A1), write(' '), write(A2),
    write(' is a valid operator.'), nl, !, opcode([A1,A2],N,O),
    write('It should be done '), write(N),
    (integer(N), write(' times.') ; write('.')), asserta(qual_error(O)),
    opcode([A1,A2],N,O2), checkretract(missed_once), nl, nl,
    writechar(*,79), nl,nl.

least_common_op(S,G,O,O2,G2,G) :- once_means_ends(S,G2,OL,NS),
    (not(member(O,OL)); not(member(O2,OL))), !.
least_common_op(S,G,O,O2,G2,Droot) :- difference(G2,S,D), all_achievable(S,D),
    applicable_op(D,O3), precondition(O3,G3),
    least_common_op(S,G2,O,O2,G3,Droot), !.

compare_solutions(S3,G,OL3,GS3,S2,G,OL2,GS2) :-
    once_means_ends(S3,G,OL3,GS3), once_means_ends(S2,G,OL2,GS2), !.
```

```
cache_states(S,G,[],GS) :- !.
cache_states(S,G,OL,GS) :- cached(S,G,OL,GS), !.
cache_states(S,G,OL,GS) :- cached(S2,G2,OL2,GS2), check_permutation(S,S2),
    check_permutation(G,G2), !.
cache_states(S,G,[O|OL],GS) :- asserta(cached(S,G,[O|OL],GS)),
    apply_op(O,S,NS), cache_states(NS,G,OL,GS), !.

apply_ops([],S,[S],S) :- !.
apply_ops([O|OL],S,[S|SL],NS) :- apply_op(O,S,S2), apply_ops(OL,S2,SL,NS).
apply_op(O,S,NS) :- get_deletepostcondition(O,S,DP), deleteitems(DP,S,S2),
    get_addpostcondition(O,S,AP), union(AP,S2,NS), !.
check_mainline_return(S) :- mainline_states(SL,O,OS,BO),
    check_mainline_return2(S,SL,O,OS,BO).
check_mainline_return(S).
check_mainline_return2(S,[S2|SL],O,OS,BO) :- permutemember(S,[S2]),
    !, write('You are returning to a previous state.'), nl.
check_mainline_return2(S,SL,O,OS,BO) :- permutemember(S,SL), !,
    write('Do you see now that your choice of the '), write(O),
    write(' action in the state with the facts ['), writelist(OS,state),
    write('] was not the best choice; the '), write(BO),
    write(' action would have been better.'), nl,
    retract(mainline_states(SL,O,OS,BO)).


/*****************************************************************/
/***************** NATURAL LANGUAGE OUTPUT ***********************/
/*****************************************************************/

writelist([],R) :- !.
writelist([X],R) :- !, writefact(X,R).
writelist([X,Y],R) :- !, writefact(X,R), write(' and '), writefact(Y,R).
writelist(L,R) :- writelist2(L,R).
writelist2([X],R) :- !, write('and '), writefact(X,R).
writelist2([X|L],R) :- writefact(X,R), write(', '), writelist2(L,R).
writefact(F,state) :- atom(F), write(F), write(' is true'), !.
writefact(not(F),state) :- atom(F), !, write(F), write(' is false'), !.
writefact(not(F),state) :- F=..[P,X], atom(X), !, write(X), is_form(X,IX),
    write(IX), write('not '), write(P), !.
writefact(not(F),state) :- F=..[P,X], !, writefact(X), is_form(X,IX),
    write(IX), write('not '), write(P), !.
writefact(not(F),state) :- F=..[P,X,Y], !, write(X), write(' not '),
    write(P), write(' '), write(Y), !.
```

```
writefact(F,state) :- F=..[P,X], atom(X), !, write(X), is_form(X,IX),
     write(IX), write(P), !.
writefact(F,state) :- F=..[P,X], !, writefact(X,state), is_form(X,IX),
     write(IX), write(P), !.
writefact(F,state) :- F=..[P,X,Y], !, write(X), write(' '),
     write(P), write(' '), write(Y), !.
writefact(F,precond) :- atom(F), write(F), write(' must be true'), !.
writefact(not(F),precond) :- atom(F), !, write(F), write(' must be false'), !.
writefact(not(F),precond) :- F=..[P,X], atom(X), !, write(X),
     write(' must not be '), write(P), !.
writefact(not(F),precond) :- F=..[P,X], !, writefact(X,state),
     write(' must not be '), write(P), !.
writefact(not(F),precond) :- F=..[P,X,Y], !, write(X), write(' must not be '),
     write(P), write(' '), write(Y), !.
writefact(F,precond) :- F=..[P,X], atom(X), !, write(X),
     write(' must be '), write(P), !.
writefact(F,precond) :- F=..[P,X], !, writefact(X,state),
     write(' must be '), write(P), !.
writefact(F,precond) :- F=..[P,X,Y], write(X), write(' must be '), write(P),
     write(' '), write(Y), !.
writefact(F,op) :- write(F), !.
writefact(F,R) :- write(F).

is_form(X,' is ') :- not(atom(X)), !.
is_form(X,' are ') :- name(X,NX), last(NX,115), !.
is_form(X,' is ').

/*********************************************************************/
/**************** ORIGINAL MEANS-ENDS PROGRAM ****************/
/*********************************************************************/

once_means_ends(State,Goal,Oplist,Goalstate) :-
     means_ends(State,Goal,Oplist,Goalstate),
     cache_states(State,Goal,Oplist,Goalstate), !.
means_ends(State,Goal,Oplist,Goalstate) :-
     means_ends2(State,Goal,Oplist,Goalstate,[]), writedebug7.
means_ends2(State,Goal,Oplist,Goalstate,Stack) :-
     cached(State2,Goal2,Oplist,Goalstate), check_permutation(Goal,Goal2),
     check_permutation(State,State2), !, writedebug6(Stack), !.
means_ends2(State,Goal,Oplist,Goalstate,Stack) :- member([State,Goal],Stack),
     !, writedebug4(Stack), fail.
means_ends2(State,Goal,[],State,Stack) :- difference(Goal,State,[]), !.
```

```
means_ends2(State,Goal,Oplist,Goalstate,Stack) :- difference(Goal,State,D),
    applicable_op(D,Operator), precondition(Operator,Prelist),
    all_achievable(State,Prelist), writedebug1(D,Operator,Stack),
    means_ends2(State,Prelist,Preoplist,Prestate,[[State,Goal]|Stack]),
    writedebug2(Prestate,D,Operator,Stack),
    get_deletepostcondition(Operator,Prestate,Deletepostlist),
    deleteitems(Deletepostlist,Prestate,Prestate2),
    get_addpostcondition(Operator,Prestate,Addpostlist),
    union(Addpostlist,Prestate2,Postlist),
    means_ends2(Postlist,Goal,Postoplist,Goalstate,[[State,Goal]|Stack]),
    writedebug3(Goalstate,Operator,Stack),
    append(Preoplist,[Operator|Postoplist],Oplist).
means_ends2(State,Goal,Oplist,Goalstate,Stack) :-
    writedebug5(State,Goal,Stack), !, fail.



/*****************************************************************/
/********************* DEBUGGING TOOLS ***********************/
/*****************************************************************/

writedebug1(D,O,Stack) :- not(debugflag), !.
writedebug1(D,O,Stack) :- length(Stack,Nm1), N is Nm1+1, write('>>Operator '),
    write(O), write(' suggested at level '), write(N), nl,
    write('to achieve difference of ['), writelist(D,state), write(']'), nl, !.
writedebug2(S,D,O,Stack) :- not(debugflag), !.
writedebug2(S,D,O,Stack) :- length(Stack,Nm1), N is Nm1+1,
    write('>>Operator '), write(O), write(' applied at level '), write(N),
    nl, write('to reduce difference of ['), writelist(D,state), write(']'),
    nl, write('in state in which '), writelist(S,state), nl, !.
writedebug3(S,O,Stack) :- not(debugflag), !.
writedebug3(S,O,Stack) :- length(Stack,Nm1), N is Nm1+1, write('>>Level '),
    write(N), write(' terminated at state in which '), writelist(S,state), nl, !.
writedebug4(Stack) :- not(debugflag), !.
writedebug4(Stack) :-
    write('>>>>Reasoning found a potential infinite loop at level '),
    length(Stack,Nm1), N is Nm1+1, write(N), nl, !.
writedebug5(State,Goal,Stack) :- not(debugflag), !.
writedebug5(State,Goal,Stack) :- write('>>>>Unsolvable problem at level '),
    length(Stack,Nm1), N is Nm1+1, write(N), nl, write('for state '),
    writelist(State,state), nl, write('and goal '), writelist(Goal,state), nl, !.
writedebug6(Stack) :- not(debugflag), !.
writedebug6(Stack) :- write('>>>>Previously computed solution used at level '),
    length(Stack,Nm1), N is Nm1+1, write(N), nl, !.
```

```
writedebug7 :- not(debugflag), !.
writedebug7 :- nl, !.

+writetrace(State,Goal) :- not(traceflag), !.
+writetrace(State,Goal) :- writechar(-,25), nl, write('Goal is: '), nl, writelistalt(Goal),
    nl, write('State is: '), nl, writelistalt(State), nl, writechar(-,25), nl, !.



*** Problem definition errors  ***


check_obvious_errors :- setof([M,A],obvious_error(M,A),MAL), !, writepairlist(MAL).

obvious_error('precondition fact missing for operator ',O) :-
    recommended(D,O), not(precondition(O,L)).
obvious_error('deletepostcondition fact missing for operator ',O) :-
    recommended(D,O), not(get_deletepostcondition(O,S,L)).
obvious_error('addpostcondition fact missing for operator ',O) :-
    recommended(D,O), not(get_addpostcondition(O,S,L)).
obvious_error('"recommended" fact missing for operator ',O) :-
    precondition(O,L), not(recommended(D,O)).
obvious_error('"recommended" fact missing for operator ',O) :-
    get_deletepostcondition(O,S,L), not(recommended(D,O)).
obvious_error('"recommended" fact missing for operator ',O) :-
    get_addpostcondition(O,S,L), not(recommended(D,O)).

issue_warnings :- setof([M,A],possible_error(M,A),MAL), !,
    write('Warnings:'), nl, writepairlist(MAL), nl.
issue_warnings.

possible_error('This fact is not creatable: ',F) :- precondition(O,PL),
    backtracking_member(F,PL), uncreatable(F).

writepairlist([]).
writepairlist([[X,Y]|L]) :- write(X), write(Y), nl, writepairlist(L).
```

```
/*************************************************************************/
/********************  MEANS-ENDS UTILITIES  *************************/
/*************************************************************************/

uncreatable(F) :- precondition(O,L), backtracking_member(F,L),
    not(in_postcondition(F)).

in_postcondition(not(F)) :- any_deletepostcondition(O,DPL), member(F,DPL).
in_postcondition(not(F)) :- randsubst(O,RSL), member([F,X,Y,Z],RSL).
in_postcondition(F) :- not(F=..[not,P]), any_addpostcondition(O,APL),
    member(F,APL).
in_postcondition(F) :- not(F=..[not,P]), any_addpostcondition(O,APL),
    member([X,F,Y,Z],RSL).

any_deletepostcondition(O,L) :- deletepostcondition(O,C,L).
any_deletepostcondition(O,L) :- deletepostcondition(O,L).

any_addpostcondition(O,L) :- addpostcondition(O,C,L).
any_addpostcondition(O,L) :- addpostcondition(O,L).

get_deletepostcondition(O,S,L) :- deletepostcondition(O,C,L),factsubset(C,S),!.
get_deletepostcondition(O,S,L) :- deletepostcondition(O,L).

get_addpostcondition(O,S,L) :- addpostcondition(O,C,L), factsubset(C,S), !.
get_addpostcondition(O,S,L) :- addpostcondition(O,L).

applicable_op(D,O) :- recommended(D2,O), subset(D2,D).
```

```
{ partial file: utilities }

/*************************************************************************/
/******************  RANDOMNESS HANDLER  *********************************/
/*************************************************************************/

do_randsubst(O,S,NS) :- randsubst(O,RL), !, do_randsubst2(RL,S,NS).
do_randsubst(O,S,S).
do_randsubst2([],S,S).
do_randsubst2([[F,NF,P]|L],S,NS) :- random(1000,K), P1000 is P*1000,
    K=<P1000, changestate(F,NF,S,S2), !, do_randsubst2(L,S2,NS).
do_randsubst2([[F,NF,P,M]|L],S,NS) :- random(1000,K), P1000 is P*1000,
    K=<P1000, changestate(F,NF,S,S2), !, write(M), nl, do_randsubst2(L,S2,NS).
do_randsubst2([C|L],S,NS) :- do_randsubst2(L,S,NS).

changestate(none,NF,S,[NF|S]) :- !, not(member(NF,S)), nl, !.
changestate(F,none,S,S2) :- !, member(F,S), delete(F,S,S2), nl, !.
changestate(F,NF,S,[NF|S3]) :- !, member(F,S), delete(F,S,S3), nl, !.

permutation([],[]) :- !.
permutation(L,[I|PL]) :- randitem(L,I), delete(I,L,L2), permutation(L2,PL).

randitem(L,I) :- length(L,N), random(N,K), item(K,L,I).

randinit :- C is cputime*1000, FC is floor(C), S is FC mod 2311,
    uniqueassert(randseed(S)).

random(N,K) :- randseed(S), nextrand(S,NS), K is NS mod N,
    retract(randseed(S)), asserta(randseed(NS)).

nextrand(S,NS) :- FIC is 100*cputime, IC is floor(FIC),
    S2 is ((S*S)+IC) mod 2311, S3 is (S2*S2) mod 2311, NS is (S3*S) mod 2311.

item(K,[],I) :- !, fail.
item(K,[X|L],X) :- K=<1, !.
item(K,[X|L],Y) :- Km1 is K-1, item(Km1,L,Y).
```

```
/*********************************************************************/
/*********************    TIMER    ******************************/
/*********************************************************************/

get_time(T) :- system("rm datetemp"), system("date >datetemp"),
     see(datetemp), get0(D1), get0(D2), get0(D3), get0(D4), get0(D5),
     get0(D6), get0(D7), get0(D8), get0(D9), get0(D10), get0(D11),
     get0(D12), get0(D13), get0(D14), get0(D15), get0(D16), get0(D17),
     get0(D18), get0(D19), seen, digitascii(D15,N1), digitascii(D16,N2),
     digitascii(D18,N3), digitascii(D19,N4), T is (N1*600)+(N2*60)+(N3*10)+N4.

digitascii(D,N) :- N is D-48.


/*********************************************************************/
/******************** UTILITY PREDICATES ***************************/
/*********************************************************************/

delete_uncreatable([],[]).
delete_uncreatable([X|L],M) :- uncreatable(X), !, delete_uncreatable(L,M).
delete_uncreatable([X|L],[X|M]) :- delete_uncreatable(L,M).

all_achievable(S,G) :- difference(G,S,D), not(unachievable_member(D)).

unachievable_member(D) :- backtracking_member(F,D), uncreatable(F).

difference([],S,[]).
difference([not(P)|G],S,G2) :- not(singlemember(P,S)), !, difference(G,S,G2).
difference([P|G],S,G2) :- singlemember(P,S), !, difference(G,S,G2).
difference([P|G],S,[P|G2]) :- difference(G,S,G2).

subset([],L).
subset([X|L],L2) :-singlemember(X,L2), subset(L,L2).

factsubset([],L).
factsubset([not(P)|L],L2) :- not(singlemember(P,L2)), !, factsubset(L,L2).
factsubset([not(P)|L],L2) :- !, fail.
factsubset([P|L],L2) :- singlemember(P,L2), factsubset(L,L2).

member(X,L) :- singlemember(X,L).

singlemember(X,[X|L]) :- !.
singlemember(X,[Y|L]) :- singlemember(X,L).
```

```
append([],L,L).
append([X|L],L2,[X|L3]) :- append(L,L2,L3).

union([],L,L).
union([X|L1],L2,L3) :- singlemember(X,L2), !, union(L1,L2,L3).
union([X|L1],L2,[X|L3]) :- union(L1,L2,L3).

deleteitems([],L,L).
deleteitems([X|L],L2,L3) :- delete(X,L2,L4), deleteitems(L,L4,L3).

delete(X,[],[]).
delete(X,[X|L],M) :- !, delete(X,L,M).
delete(X,[Y|L],[Y|M]) :- delete(X,L,M).

checkretract(S) :- call(S), retract(S), !.
checkretract(S).

check_permutation(L,M) :- subset(L,M), subset(M,L), !.

subsequence([],L) :- !.
subsequence([X|L],[X|M]) :- !, subsequence(L,M).
subsequence(L,[X|M]) :- subsequence(L,M).

permutemember(X,[X|L]) :- !.
permutemember(X,[Y|L]) :- subset(X,Y), subset(Y,X), !.
permutemember(X,[Y|L]) :- permutemember(X,L).

last([X],X).
last([X|L],Y) :- last(L,Y).

elimdups([],[]).
elimdups([X|L],M) :- singlemember(X,L), !, elimdups(L,M).
elimdups([X|L],[X|M]) :- elimdups(L,M).

uniqueassert(Q) :- Q=..[P|L], length(L,N), abolish(P,N), asserta(Q).

backtracking_member(X,[X|L]).
backtracking_member(X,[X|L]) :- backtracking_member(X,L).

helpword(help).
helpword(h).
helpword(huh).
```

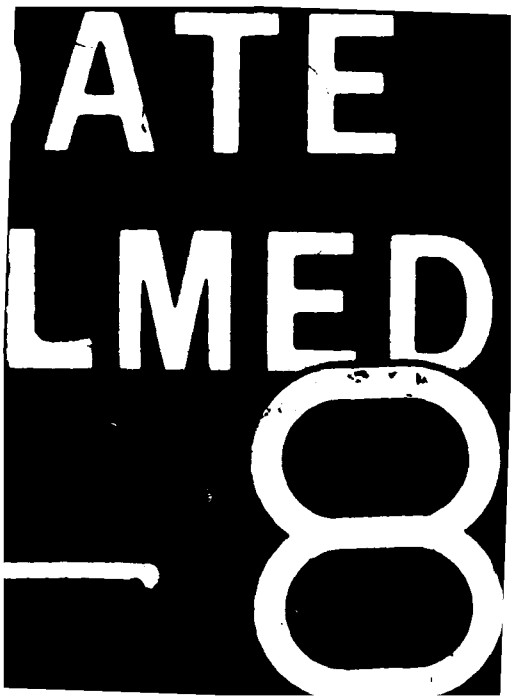
LIST OF REFERENCES

1. American Red Cross, *American Red Cross: Adult CPR*, 1987.
2. *The Journal of the American Medical Association*, v. 255, no. 21, pp. 2841-3044, 1986.
3. Kearsley, G. P., *Artificial Intelligence & Instruction*, Addison-Wesley, 1987.
4. K. L. Zinn, "Computer-Assisted Learning and Teaching," in *Encyclopedia of Computer Science and Engineering*, ed. A. Ralston, pp. 294-302, Van Nostrand Reinhold, New York, 1983.
5. Steinberg, E. R., *Teaching Computers to Teach*, Lawrence Erlbaum Associates, 1984.
6. Barr, A. and Feigenbaum, E. A., *The Handbook of Artificial Intelligence*, v. 2, William Kaufmann, 1982.
7. Woolf, B. and Cunningham, P. A., "Multiple Knowledge Sources in Intelligent Teaching Systems," *IEEE Expert*, pp. 41-54, Summer 1987.
8. Wenger, E., *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann Publishers, 1987.
9. Sleeman, D. and Brown, J. S., *Intelligent Tutoring Systems*, Academic Press, 1982.
10. Freedman, R. S. and Rosenking, J. P., "Designing Computer-Based Training Systems: OBIE-1:KNOBE," *IEEE Expert*, pp. 31-38, Summer 1986.
11. Merrill, M. D., "An Expert System for Instructional Design," *IEEE Expert*, pp. 23-37, Summer 1987.
12. Rowe, N. C., *Artificial Intelligence Through Prolog*, Prentice-Hall, 1988.
13. Clocksin, W. F. and Mellish, C. S., *Programming in Prolog*, 2nd ed., Springer-Verlag, 1984.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chief of Naval Operations Director, Information Systems (OP-945) Navy Department Washington, D.C. 20350-2000	1
4.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93943-5000	2
5.	Curriculum Officer, Code 37 Computer Technology Naval Postgraduate School Monterey, California 93943-5000	1
6.	Professor Neil C. Rowe, Code 52Rp Computer Science Department Naval Postgraduate School Monterey, California 93943-5000	2
7.	Commanding Officer Naval Research Laboratory Washington, DC 20375	1
8.	Chief of Naval Education and Training Naval Air Station Pensacola Pensacola, Florida 32508	1

9. LCDR Debra S. Campbell, USN 3
Computer Services, Stop 2a
Ward Hall
U. S. Naval Academy
Annapolis, Maryland 21402


ATE
LMED
8