BOB FISHER

DEPARTMENT OF ARTIFICIAL INTELLIGENCE
UNIVERSITY OF EDINBURGH

DAI Working Paper No. 121
Date: August 1982

Title: The Potential of Expert System Based Training Aids

Authors: Robert B. Fisher and James A. M. Howe

Abstract:

The emerging field of knowledge engineering is involved with the development of computer based expert systems, whose problem solving capabilities are intended to rival those of human experts. The development of an expert system requires the acquisition and formal encoding of the large quantities of both factual and procedural knowledge associated with the specific domain of expertise. It would be ideal if this knowledge could then be used, in conjunction with the expert system, to help train humans to greater proficiency levels. This paper addresses the problem of creating expert training systems. It starts with a background discussion of research on both expert and educational systems. Then, a speculative discussion follows on the nature of reasonable training systems and how they might be developed and used. The paper also includes discussion of specific points raised by the funding agency, and some recommendations on the most fruitful course for further work.

Acknowledgements:

The authors gratefully acknowledge the funding of the Manpower Services Commission/Training Services Division (Contract "Artificial Intelligence Research in Expert Systems and its relevance to High Level Training"). They would also like to thank H. Pain, R. O'Keefe and P. Ross for having taken the time to offer their advice and opinions on this paper and its subject.

Table Of Contents

1. Introduction

Expertise is scarce. To become an expert takes years of experience; to find it (in others) requires extensive search; to hire it (when available) is expensive. Yet, demands for it are ever increasing, as science, technology and commerce become more and more complex. The development of computer-based expert systems, a product combining artificial intelligence research with specific areas of expertise, is one attempt to reduce this scarcity.

Building an expert system depends crucially on the ability to acquire from a group of specialists, familiar with a particular area, the factual and judgemental knowledge needed to solve the problems of the domain. This knowledge is then embedded into a specially constructed computer program, which attempts to use the knowledge to solve problems at an expert's performance level. Typical areas investigated to date, where the system's performance approaches expert level, include specialized medical diagnosis, chemical structure analysis and mineral exploration.

But, these systems must still be regarded as prototypes because, unlike human experts, they can only handle narrow, restricted domains. For this reason alone, we will need human experts for the forseeable future.

Expert systems can only be created and extended by human experts. Further, the cost of building and operating expert systems is significant, and hence will tend to restrict their use to large corporations or to specialized institutions, whereas much innovative activity is carried out by small organizations. So, the hard problem is how to make expertise available to them at relatively low cost. One possible solution is to try to establish a symbiotic relationship between the user and the system, so that most of the decision making is made by human beings, with the expert programs providing training.

We can envisage training taking a number of different forms, depending upon the needs of the user. First of all, there can be a need for expert guidance. For example, once per year a user might need to use a complex technique, such as finite element analysis of a mechanical design. Having access to a program which could advise him/her how to best apply the technique to the immediate problem confers obvious benefits. The expert system, **SACON** (Ben79), was designed for just such a purpose.

A second need is that of sharpening up existing knowledge, as learning to apply previously acquired knowledge to complex tasks. For example, electronic trouble-shooting techniques are often taught formally, in isolation from the equipment. Having access to a program which could introduce faults into an apparatus, and then call for principled and verifiable explanations from the student would also confer obvious benefits. One can envisage SOPHIE (Bro75) as an example of an expert system which does this.

The third need is to train people from scratch, particularly in crucial areas where existing training methods are poor. This is a more daunting task than the other two, since it would be necessary to create a computer-based tutor which embodies the attributes of the ideal human tutor, such as understanding the needs, abilities and current development of

the learner, having a thorough understanding of the domain, and having a firm grasp of pedagogical methods. An example of a system attempting to satisfy this specification in part is Clancey's GUIDON program for teaching medical students to diagnose infections, prescribe theraples and discuss their reasoning (Cla79c).

This paper examines the concept of the expert system as a training aid, motivated by a discussion of research on expert and intelligent instructional systems. We discuss ideal systems, and then back off to discuss feasible systems. Also included is a discussion of the practical considerations of development and deployment.

2. Expert System Background Discussion

This section provides some background on the nature of expert systems. It gives an overview of the types of expert systems, the techniques used in their construction, and some specific example systems.

2.1 What is an expert system?

An expert system, otherwise known as a consultation program, duplicates the reasoning process of a human expert, forming judgements, evaluating alternatives and making decisions. It is constructed by interviewing human experts and encoding their reports of how they perform their skill, eg. diagnosing a disease or interpreting scientific data, in some knowledge representation, such as a set of situation => action rules that operate in a relational network. These rules state that if such-and-such a condition holds, then such-and-such possibilities are likely (or such-and-such action should be taken). Below is an example of such a rule, taken from the **MYCIN** system (Sho76) which diagnoses bacterial blood infections and meningitis infections, and recommends drug treatments:

RULE 85

IF: (1) the site of the culture is blood, and
(2) the gram stain of the organism is gramneg, and
(3) the morphology of the organism is rod, and
(4) the patient is a compromised host

THEN: There is suggestive evidence (0.6) that the identity of the organism is pseudomonas-aeruginosa

Rules may also govern the use of other rules. Below is an example taken from TEIRESIAS (Dav77b), a system with which an expert diagnostician interacts to build up the knowledge base used by MYCIN:

META-RULE

IF: (1) the patient is a compromised host, and
(2) there are rules which mention in their premise Pseudomonas, and
(3) there are rules which mention in their premise Klebsiellas

THEN: There is suggestive evidence that the former rules should be done before the latter

This method of representing human skill is versatile. It can capture "fuzzy" knowledge and rules-of-thumb, as well as purely algorithmic knowledge (though it is most effective for representing knowledge needed for independent, single-step deductions. Further, the relationship between "fuzzy" knowledge and reasoning is not well understood.) If the "grain size" of the knowledge and the problem decomposition are chosen properly, production rules lead to easier modifications, where rules are either changed, inserted or deleted. This enables the system to develop its expertise in stages, without the need to re-construct the knowledge base.

Besides a data-base of information and a set of rules specific to a particular skill, each system is equipped with an inference system. An example of this type of system is MYCIN's simple backward-chaining inference system. It operates by setting up a goal, and by retrieving from its data-base the list of rules whose conclusions bear on this goal. If, in the course of evaluating the premise of one of these rules, some other piece of information is needed which is not yet known, the system sets up a sub-goal to find out that information. This, in turn, causes other rules to be applied, and so on. Questions may be asked during the consultation when the rules fail to deduce the necessary information. If the user cannot supply the data, the rules requiring it will fail. Thus, the rules unwind to produce a succession of goals, and it is the attempt to achieve each goal that drives the consultation. Conceptually, the backwards-chaining inference system implements a generate-and-test strategy, where the legal move generators are based upon logical rules of inference, and the tests take a variety of forms varying from simple tests (MYCIN: Is the organism aerobic?) to complex heuristic evaluations (MOLGEN (Fri79): Will this reaction actually take place?).

This illustrates one major distinction between expert systems and other significant computer programs: the expert systems embody facts specific to a domain, rules for making deductions, and a reasoning system.

An early expert system is MACSYMA (Mos71), a program to aid physicists and applied mathematicians in such tasks as solving equations, doing symbolic integration or factorizing polynomials. Its abilities are limited to correctly performing mathematical transformations requested by the user, unlike later systems which also reason about which actions should by taken. Probably the most famous early system is DENDRAL (Buc78), which infers

chemical structure from organic mass spectrogram and nuclear magnetic resonance data. For some families of molecules, particularly estrogenic steroids, it operates more accurately and much more quickly than the best human mass-spectrum analyst. Several programs which have been developed for medical diagnosis and treatment are: **MYCIN**, **PUFF** (HPP80) which diagnoses pulmonary function disorders, and **HEADMED** (van79) which diagnoses a range of psychiatric disorders and can recommend treatment. Other systems include: **PROSPECTOR** (Dud79), a system for helping a geologist evaluate the mineral potential of an exploration site; **GAMMA** (Bar79), a system for interpreting gamma-ray activation spectra to identify the elemental composition of unknown substances; and **SACON** (Ben79), a system for providing advice about how to use a complex Finite-Element structural analysis program.

2.2 A brief look at several expert systems

MYCIN

The **MYCIN** system, described by Davis, Buchanan and Shortliffe (Dav75), is one of the first successful expert system programs. It diagnoses and provides advice about the diagnosis and treatment of bacteremia and meningitis at consultant performance levels. As an elementary model of an expert system, of how knowledge is to be collected and used, and of what facilities are needed in such a system, it remains unsurpassed. (Though more sophisticated systems have since been developed.)

The study of **MYCIN** is especially relevant to the tutorial concerns of this paper: its knowledge is largely judgemental, yet sufficiently precise as to provide expert level performance.

The overall goal of a **MYCIN** consultation is to identify the organisms responsible for a patient's illness, and to suggest treatments. **MYCIN** accomplishes this by receiving the patient data through an interactive dialogue, then attempting to deduce the responsible organisms, pausing to ask questions when appropriate (much like a human consultant). Finally, when the diagnosis is complete, these results and a suggested therapeutic regimen are returned to the user. The user can then ask questions of the system, ascertaining how conclusions were reached and why particular facts were relevant. Further, the user can ask general questions about the system's medical knowledge. Figures 2 and 3 show portions of a session with **MYCIN**. They are taken from a paper by Davis et al (Dav75).

The **MYCIN** system consists of six major components:

consultation program - receives case data, deduces organisms, suggests treatments

explanation program - provides explanations of how particular conclusions were reached, and why particular actions were chosen. This covers both successful and unsuccessful reasoning, but only about the current case.

question answerer - allows simple natural language questions about **MYCIN**'s medical knowledge base

knowledge acquisition program - allows the expert to add new rules and modify old ones currently used by **MYCIN**. This is especially important considering that medical knowledge is constantly changing.

knowledge base - the set of medical facts and methods used by **MYCIN**, recorded as a set of production rules and data tables

patient data base - the initial case data, all deduced information and the records of the system's reasoning about the case

The medical model used by **MYCIN** is fairly simple: a patient has one or more infections, at particular sites, from which cultures can be made, which contain particular organisms. Each object (ie., organism, infection or drug) has a variety of attributes, such as portal of entry, infection type, or drug sensitivity.

There are two types of reasoning used during the consultation. In the diagnosis phase **MYCIN** uses a goal-directed form of reasoning in which the system attempts to deduce which organisms were responsible for the

observed infection, by reasoning backward to the evidence necessary to support such a conclusion. These deductions are recorded in an AND/OR tree, where the AND records the factors simultaneously necessary, and the OR records the alternative hypotheses. In the treatment phase, a generate and test paradigm is used, in which potential therapies are tested against the set of organisms, so as to select the best overall therapy.

The reasoning used by the physician is only approximate. Correspondingly, the reasoning of MYCIN associates a certainty factor (varying from -1 to +1) with each deduction. These factors express the degree of belief in the conclusion (from strongly against to strongly for).

The medical knowledge consists of about 450 production rules, encapsulating the reasoning used by the expert, and tables of medical facts. One such rule is shown in figure 1.

IF: (1) the infection is primary-bacteremia, and
(2) the site of the culture is one of the sterilesites, and
(3) the suspected portal of entry of the organism is the gastro-intestinal tract,

THEN: there is suggestive evidence (0.7) that the identity of the organism is bacteroides

Figure 1 - a sample MYCIN production rule (Dav75)

- - - - - - - - PATIENT-1 - - - - - - - -
(1) Patient's name: (first-last)
**FRED SMITH

{*background data*}

(2) Sex:
**MALE
(3) Age:
**55
(4) Have you been able to obtain positive cultures from a site at which Fred Smith has an infection?
**YES
- - - - - - - - INFECTION-1 - - - - - - - -
(5) What is the infection?
**PRIMARY-BACTEREMIA
(6) Please give the date and approximate time when signs or symptoms of the primary-bacteremia (INFECTION-1) first appeared. (mo/da/yr)
**MAY 5, 1975
The most recent positive culture associated with the primary-bacteremia (INFECTION-1) will be referred to as:
- - - - - - - - CULTURE-1 - - - - - - - -
(7) From what site was the specimen for CULTURE-1 taken?
**BLOOD
(8) Please give the date and time when this blood culture (CULTURE-1) was obtained. (mo/da/yr)
**MAY 9, 1975
The first significant organism from this blood culture (CULTURE-1) will be referred to as:
- - - - - - - - ORGANISM-1 - - - - - - - -
(9) Enter the identity of ORGANISM-1:
**UNKNOWN

{*typically the identity is not known and must be inferred*}

(10) Is ORGANISM-1 a rod or coccus (etc.):
**ROD
(11) The gram stain of ORGANISM-1:
**GRAMNEG
(12) Have there been POSITIVE cultures from sites which are NOT associated with infections about which you are seeking advice?
**NO
(13) Did ORGANISM-1 grow aerobically?
**YES (8)
(14) Did ORGANISM-1 grow anaerobically?
**YES
(15) What is the suspected portal of entry of ORGANISM-1 into this sterile site?
**GI (6)
...

{*more questions follow, to gather sufficient information to infer possible identities of the infecting organisms*}

FIG. 1. Initial segment of a session with the consultation system. User responses are in boldface and follow the double asterisks. Italicized comments at the right are not part of the actual dialog.

Artificial Intelligence 8 (1977), 15–45

Figure 2 - a sample **MYCIN** case dialog (Dav75)

My therapy recommendation will be based on the following probable infection(s) and potential causative organism(s):

INFECTION-1 is PRIMARY-BACTEREMIA

{possible identities}

⟨Item 1⟩ The identity of ORGANISM-1 may be PSEUDOMONAS-AERUGINOSA
⟨Item 2⟩ The identity of ORGANISM-1 may be KLEBSIELLA-PNEUMONIAE
⟨Item 3⟩ The identity of ORGANISM-1 may be E. COLI
⟨Item 4⟩ The identity of ORGANISM-1 may be BACTEROIDES-FRAGILIS
⟨Item 5⟩ The identity of ORGANISM-1 may be ENTEROBACTER
⟨Item 6⟩ The identity of ORGANISM-1 may be PROTEUS-NON-MIRABILIS

(46) On a scale of 0 to 4, where higher numbers indicate increasing severity how would you rate Fred Smith's degree of sickness?
**3
(47) Does Fred Smith have a clinically significant allergic reaction to any antimicrobial agent?
**NO
(48) Fred Smith's weight in kilograms (or ⟨number⟩ POUNDS):
**70

{note that 2 drugs cover for 6 potential organisms}

[Rec 1] My preferred therapy recommendation is as follows:
In order to cover for Items ⟨1 2 3 5 6⟩:
Give: GENTAMICIN
Dose: 119 mg (1.7 mg/kg) q8h IV [or IM] for 10 days
Comments: Modify dose in renal failure
In order to cover for Item ⟨4⟩:
Give: CLINDAMYCIN
Dose: 595 mg (8.5 mg/kg) q6h IV [or IM] for 14 days
Comments: If diarrhea or other GI symptoms develop, patient should be evaluated for possible pseudomembranous colitis.

Do you wish to see the next choice therapy?

{alternative choices are available}

**Y
Which, if any, of the above drugs do you wish eliminated from consideration?
**CLINDAMYCIN

[Rec 2] Next best therapy recommendation:
In order to cover for Items ⟨2 3 4 5 6⟩:
Give: CHLORAMPHENICOL
Dose: 1 gm (15 mg/kg) q6h IV for 14 days
Comments: Monitor patient's white count
In order to cover for Item ⟨1⟩:
Give: GENTAMICIN
Dose: 119 mg (1.7 mg/kg) q8h IV [or IM] for 10 days
Comments: Modify dose in renal failure

FIG. 2. Final segment of a consultation. Alternative choices for therapies are generated if requested.

Artificial Intelligence 8 (1977), 15–45

Figure 3 – a sample MYCIN diagnosis and therapy selection (Dav75)

R1

The R1 system, as described by McDermott (McD80), configures computer systems. At present, it is used to configure only one type of machine, the VAX-11/780, a large mini-computer system built by Digital Equipment Corporation. This expert system is discussed as an example of a system in actual use.

R1 receives, as input, a customer order, which is a list of required equipment. It takes this order and produces, as output, a configuration map which directs the technicians who assemble the system. The output description includes components which are required, but were omitted from the specification.

The system uses a model which almost exactly corresponds to a real VAX-11/780 system. It consists of a set of components, whose descriptions are reduced to a set of relevant features (attribute/value pairs). Solving the problem consists of assigning connections to other devices and device locations in the final structure.

The reasoning model is as simple. It depends upon a specialized problem solving context in which any action appropriate in the current context is guaranteed to get the problem closer to a solution. So, R1 models the problem as a series of descriptions of the current status of the design (states), and its problem solving consists of applications of its rules, each of which moves the system one state closer to a solution. (There is one exception, which can temporarily generate spurious configurations.) The knowledge of the system is encoded as a set of production rules (772, of which 666 relate to the actual problem). A typical solution requires about 800 sequential states.

An example of a rule used by the system is (English translation):

ASSIGN-UB-MODULES-EXCEPT-THOSE-CONNECTING-TO-PANELS-4

IF: the current context is assigning devices to unibus modules
AND there is an unassigned dual port disk drive
AND the type of controller it requires is known
AND there are two such controllers neither of which has any devices assigned to it
AND the number of devices that these controllers can support is known

THEN: assign the disk drive to each of the controllers
AND note that the two controllers have been associated and each supports one device

The system took about 1 man-year of a knowledge engineer's time to develop, plus what appears to be about 3 man-months of expert's time. The system was validated by correctly generating 50 system configurations taken from real orders. The system is currently in actual use and has generated over 500 configurations. Generation of a configuration takes about 2.5 minutes of cpu time (about 5 minutes delay).

No description of the user interface is available: it is presumed to be

just a prompted dialogue, which inputs the required equipment list. No mention has been made of any question-answering facility, which probably isn't necessary in this case.

GAMMA

The **GAMMA** system, as described by Barstow (Bar79), analyzes gamma-ray activation spectra. It produces a description of the concentration of elements in a chemical sample. The analysis process involves bombarding the sample with a neutron beam, and examining the gamma-rays emitted when the atoms in the sample decay. The input into **GAMMA** is the detected gamma-ray spectrum.

GAMMA uses a deep domain model, which consists of six levels of description. The levels are:

1. elements in the original sample
2. isotopes in the original sample
3. isotopes after neutron bombardment
4. decays of unstable isotopes
5. gamma-ray emissions during decay
6. gamma-ray detections during decay

Each of the levels consists of a description of the hypothesized entities (ie., elements, isotopes) and concentrations appropriate to the level, and links to the entities at the next lower level. The system knowledge consists of the rules of physics which describe how entities at one level relate to those at the next lower level.

The reasoning of the system proceeds in two stages. The first stage produces candidate elements and concentrations by hypothesizing elements at the top level, using the domain database, to deduce spectra at level 6, and then comparing the predicted spectra to the observed spectra. The second stage generates the subset which best describes the input spectra. This is done by a specialized optimization algorithm, and is of little general interest here.

The major domain expertise is the set of the facts and formulas relating the different model levels. These are highly technical, and are generally obtained from books and scientific tables when needed.

No details of the actual system performance have been given, other than that its accuracy approaches expert levels.

It appears likely that the system uses a very data-oriented input interface (perhaps a file with spectral data) and has no question answering facility. Further, the reasoning methods appear to be fairly domain specific.

SACON

The SACON system, described by Bennett and Engelmore (Ben79), gives advice on the use of a complex structural analysis program (MARC).

The MARC program is designed to simulate and analyze mechanical structures such as aircraft wings, reactor pressure vessels, bridges and buildings. It has a large variety of user-selectable options covering analysis methods, object geometries and object materials. The user's goal in a consultation with MARC is to correctly analyze the structure, with necessary accuracy and for minimal cost. The problem with its use stems from the variety of options - the typical structural engineer knows what it is he/she wants done, but not how. Bennett and Engelmore comment that it requires an average of one year's use of the system to gain this experience.

The goal of the engineer's consultation with SACON is to receive instructions on which of MARC's options will result in a correctly analyzed structure. After a description of the structure is entered into SACON, the system proceeds to do a preliminary analysis of the structure, based upon its knowledge of structures, materials, loadings, stress and critical behaviors. SACON then suggests the factors and types of behavior that need more detailed analysis by MARC, and the options needed to provide the analysis. The engineer then uses the MARC program to complete the analysis.

SACON is built upon the EMYCIN domain-independent expert system (see section 5.5). Like MYCIN, it uses a goal-directed reasoning process (whose top-level goal is to deduce an analysis strategy based upon worst-case stress and deflection behaviors). This reasoning, however, did not need to use the "certainity factor" facilities of EMYCIN. Its knowledge is contained in a set of approximately 170 production rules. As the system is built upon the EMYCIN base, it has available the explanation, question answering and knowledge acquisition facilities.

Bennett and Engelmore comment that the development of SACON was very easy and required minimal changes to EMYCIN. Further, they note that the experts made greatest progress in their formalization of domain knowledge (rule acquisition) when the overall structure of SACON's reasoning (ie., what it was to reason about, based upon what evidence, and when) was decided. This directly corresponds to the selection of the domain model mentioned in the previous expert system descriptions.

A typical rule is listed below:

RULE 50

IF:
1) The material composing the substructure is one of: the metals, and
2) The analysis error (in percent) that is tolerable is between 5 and 30, and
3) The non-dimensional stress of the sub-structure is greater than 0.9, and
4) The number of cycles the loading is to be applied is between 1000 and 10000

THEN: It is definite (1.0) that fatigue is one of the stress behavior phenomena in the sub-structure

2.3 A list of developed expert systems

Table 1 contains a list of the expert systems described in the public literature. Included with the name is also a reference and a brief description of its application domain and research novelty.

MEDICAL SYSTEMS

MYCIN (Sho76,van78,Dav75) - antimicrobial therapy applied to bacterial blood infections and meningitis: Heuristic/judgement rules explicit. Inexact reasoning. Explains reasoning.
PUFF (HPP80) - pulmonary disease diagnosis: Report generation.
CENTAUR (Aik79) - pulmonary disease diagnosis: Prototype structured data base guides data acquisition, controls reasoning, and orders user interactions.
VM (Fag79) - mechanical breathing apparatus management: Time-based reasoning.
ONCOCIN (HPP80) - Cancer therapy management: Planning and tracking of treatment regimens. Information collection.
RX (HPP80) - Chronic disease studies: Induction of causal relationships over large data bases.
WHEEZE (Smi80) - pulmonary disease diagnosis: Uses a frame structured database and an agenda to help order reasoning.
MDX (Cha79) - liver cholestasis diagnosis: attention focusing, diagnosis refinement
IRIS (Tri77) - medical diagnosis/glaucoma: reasoning based upon propagation of implications
PIP (Szo78) - medical diagnosis: categorical and probabilistic reasoning, frame structured knowledge base
CASNET/glaucoma (Wei77) - glaucoma diagnosis: Model of disease process.
OWL/digitalis advisor (Swa77) - digitalis therapy management: Uses procedural code for explanations.
TEIRESIAS (Dav77b) - expert on data and reasoning structures of **MYCIN**: Meta-level knowledge of rules. Explanations of reasoning. Rule base manipulation.
HEADMED (van79) - diagnosis and drug recommendations for range of psychiatric disorders: test of **EMYCIN** base

SCIENTIFIC SYSTEMS

DENDRAL (Buc78) - molecular structure elucidation from mass spectra data: Plan/generate/test paradigm, heuristic search
Components:

CONGEN (Car79) - generates plausible molecular structures.
Meta-DENDRAL (Buc78) - induces rules to explain spectral data.

MOLGEN (Fri79) - Planning molecular genetics experiments (structural analysis): Plan generation and constraint satisfaction.
CRYSALIS (Eng79) - Infers 3D protein structure from x-ray crystallographic data: Hierarchical processing model and knowledge representation.
GAMMA (Bar79) - Element analysis from gamma ray activation spectra.
PROSPECTOR (Dud79, Gas80) - mineral resource evaluation from geological models and field data
CONCHE (Chi79) - organic acid reactions: Learns rules from user.
SECS (Wip78) - design of organic synthesis reactions: interactive problem solving

ENGINEERING SYSTEMS

SACON (Ben79) - advice on use of complex structural analysis program.
R1 (McD80) - Advice on configuration of Digital VAX 11/780 computer systems to suit customer specifications.
RITA (Wat79) - programmable user agent for intelligent interface between computer system and user: restricted domain expert system frameworks
ROSIE (Wat79) - framework for expertise based simulation models: restricted domain expert system frameworks
PSI (McC80b, Kan77, Gre79) - Automatic computer program generation based upon user specifications.
LOGIN or **SU/X** (Nii78) - continuous signal interpretation (ie petroleum bore hole measurements): Geological model based.
DART (HPP80) - computer hardware fault diagnosis. Component functional models.
VLSI (HPP80) - VLSI circuit layout: Geometric models satisfying local and global constraints.
SADD (Gri80) - digital circuit design based upon functional specification
EL (Sus77) - analog circuit design
??? (Sus80) - VLSI circuit design: complex design maintenance

DOMAIN-INDEPENDENT EXPERT SYSTEM STRUCTURES

EMYCIN (van79) - provides facilities for encoding knowledge into production rules, reasoning control structure and man-machine interface: Backward reasoning, AND/OR inexact reasoning, explanations.
HEARSAY III (Bal80) - Blackboard model, multiple cooperating experts.
AGE (Nii79) - domain independent inference, control and representational techniques: expertise on constructing expert systems
XPRT (Ste79) - programming system for developing sophisticated reasoning programs: models process as a collection of communicating expert modules
RLL (Gre80) - language for knowledge representation: use of knowledge about representation techniques
EXPERT (Wei79) - medical base system: Hypothesis/finding/rule model.
AL/X (Rei80) - PROSPECTOR based Bayesian network reasoning model. Minicomputer implementation.

OTHER SYSTEMS

AM (Len77) - discovery of mathematical concepts
HEARSAY II (Les77) - speech understanding. Hierarchical blackboard acted upon by cooperating experts.
MECHO (Bun79a) - solves secondary school mechanics problems stated in English: meta-level problem solving control
LDS (Wat80) - legal analysis system. Considerations of vagueness, statute changes
TAXMAN II (McC80a) - tax law application
PONTIUS-0 (Gol77) - aircraft pilot expert: skill acquisition, representation and interaction
GUIDON (Cla79c) - teaching student medical diagnosis via case method: dialog management, explicit tutorial strategies, student modelling
MACSYMA (Mos71) - Symbolic mathematical manipulation
ADVISOR (Gen79) - consultant to **MACSYMA** system users: induction of user's plans.
PRESS (Bun79b) - algebraic manipulations, equation solving: Meta-level

selection of rewrite rules

Table 1 - Summary of existing expert systems

2.4 Some common techniques used in the construction of expert systems

This section briefly discusses techniques used in the expert systems listed in section 2.3. Though the knowledge engineering techniques discussed below are probably not of general interest, nor do they affect the educational goals, they directly affect the educational techniques feasible and practical in tutoring systems. Feigenbaum (Fei79) gives a good discussion of these topics.

Explicit knowledge representation - one of the most important features of a well constructed expert system. The word "explicit" indicates that the subject domain knowledge is not embedded in the midst of a computer program, but rather is one of the externally supplied data structures used by the program. Though this technique can degrade system response times, it offers the advantage that programs can reason about the knowledge (as well as with the knowledge) which may be fundamental to the success of the expert system. This feature is fundamental to the use of expert systems as training aids, because it is this explicit knowledge that must be taught to the student. There are several techniques for representing this knowledge, of which the most common one, production rules, is discussed below. (Other knowledge representations include problem models, predicate calculus, semantic nets, frames and data tables.) Note that, while the knowledge is explicit, this does not imply that it is represented in a form legible to a non-specialist. However, it is usually possible to convert internal forms to natural language equivalents, for study by the student.

production rules - also known as condition-action rules. These are logical statements of the form: "If conditions C1, C2, ... CN hold then do action A". In the case of most of the systems listed above, the actions are either logical (ie., conclude some additional facts) or procedural (ie., manipulate the data structures representing the current state of the problem solution). This rule format is ideal for representing most of the knowledge found in expert problem solving: it is structured, modular, of small size and formulated by a human expert as a complete rule (that is, encapsulates a complete step of reasoning), and thus represents a suitable chunk of knowledge to be taught to a student.

forward/backward reasoning - also known as data directed/goal directed reasoning. In the former case, the system starts from its input data and, after passing through intermediate states, concludes about the goals desired by the system (such as circuit layouts, chemical structures, medical therapies, etc.). Backward reasoning consists of choosing a set of goals, and reasons about which data would be needed in order to achieve those goals. If the data is not present, or contradictory, then no conclusions can be made (unless the system is designed to inquire of the user at these points). These two reasoning modes are representative of the types of reasoning procedures that can be taught to the student.

AND/OR decompositions - a problem solving technique used for reducing a problem into a set of constrained subproblems. The **AND** term refers to a set of subproblems, all of which must be solved in order to solve the main problem. The **OR** term refers to a set of alternative subproblems, one of which must be solved in order to solve the main

problem. (The terms problem and subproblem in this case refer to the current system goals, whether actions or conclusions). The production rule structure described above fits the **AND/OR** decomposition model, when used in a backward reasoning context. The multiple conditions needed to enable a rule corresponds to the **AND** case, and the consideration of alternate rules to conclude the result corresponds to the **OR** case. In terms of our tutorial system interests, this decomposition strategy is an appropriate technique for the student to learn and use, so as to reduce the complexity of the problems being solved. In this case, the **AND** term models the multiple actions that need be done to accomplish a task, whereas the **OR** term models the alternative approaches a student can take to solving the problem. (There are other suitable decompositions, such as hierarchical planning.)

search - when the expert system examines a set of alternatives to find the best solution to a problem or sub-problem. Systems which do a large amount of searching are probably not well suited for training systems, because much of their strength comes from their computational power rather than their reasoning abilities. A human expert must use some other technique (or have some additional knowledge) in order to solve the problem.

multiple cooperating experts - in which the knowledge of the system is packaged into a set of programs, each of which is an expert in a particular speciality. All contribute to the total solution. This provides the student with the notions of both suitable compartmentalizations of the knowledge and of the type of data interface needed between the specialists.

blackboard models - in which an expert system records the intermediate results of its computations in a data structure accessible to all components of the expert and tutorial system. This structure is ideal for communication between specialist modules, each of which may be expert on just particular aspects of the whole problem, and none of which need have specific knowledge of the other experts. This blackboard may have multiple levels of information, and the specialists may just access particular levels.

agenda - because humans and most computer systems can only do one thing at a time, they may need to keep an agenda of tasks which need to be done in the future. In the case of the expert systems, this agenda is a list of subproblems to be solved, or deductions to be done. Teaching the student the expert's rationale for the placement and the priority of the subtask is an appropriate training goal.

explanation of reasoning - most expert systems are capable, in a weak sense, of explaining the deductions that were made in solving a particular problem. This generally consists of the ability to recite which rules were used to support a given deduction and the purpose of the deduction. These explanations are essentially static and do not depend upon any user model.

2.5 Domain types

In this section we discuss some of the types of domain models used in existing expert systems.

The major category distinction in expert systems is between heuristic and theory driven models. A heuristic model in one whose knowledge is obtained from the experience of some human expert, and is not necessarily consistent, nor does it rely upon any underlying formal theory. The MYCIN system is an example of such a system. In this case, the diagnosis and treatment of the bacterial blood diseases is largely based upon experience mediated by some laboratory tests. The need for prompt treatment influences the methods usable, as it eliminates the more lengthy laboratory procedures. Note that the methods do not imply the non-existence of an underlying model, of biochemistry in this case. Clancey (Cla79c) uses the notion of the input/output representation of knowledge to characterize these domains - in that the expert knows the relationships that hold between causes and effects, but not the underlying mechanisms relating the two.

Theory driven models are those which are based on extensive formal models of the underlying process. The GAMMA system (Bar79) is an example of this type of system, in that the relationship between the various aspects of the model, say the elemental and isotopic descriptions, is well-defined by the laws of physics.

Often, the individual components of the model have a strong theory (such as electronic circuit elements), but the use of such components in the design process is only guided by a heuristic based theory.

The next category distinction is between shallow and deep domain models. This mainly applies to the theory driven models, as the heuristic models are generally shallow in nature. By deep models, we mean that the theory knowledge can be structured into a number of layers, with a reasonable theory linking the elements of each layer. The GAMMA (as described in section 2.2) and HEARSAY II systems are examples of two deep domain systems (more than 5 layers). (The HEARSAY II system has the following layers: speech parameters, segments, syllables, words, word-sequences, phrases).

Another distinction is between weak and strong models - that is, does the subject of expertise conform to a well-structured system. The CRYSALIS system expects such of its chemical structures, and GUIDON expects the same of its tutorial dialogs, although both have the built-in flexibility needed to capture the natural variations inherent in their domains. The MYCIN system has a weak model base, in that the major elements of treatments, infections, organisms and symptoms have only general relationships to each other.

The type of support provided by the expert system is also relevant. In the case of DENDRAL - its benefit is largely due to its computational abilities, in that the computer's speed allows it to quickly evaluate and eliminate many of the competing alternative chemical structures. In MYCIN, on the other hand, the computer provides the support by managing the complexities of reasoning over a large set of alternative hypothesis and evidence rules.

The type of reasoning is also a consideration: some systems, such as R1 (computer architecture layout), have a lengthy sequential series of

deductions with minimal consideration of alternatives, whereas the medical domain requires a short series of deductions with a large set of alternatives.

Lastly, we mention some of the general categories of computation that expert systems perform. They are:

diagnosis/deduction
    objects (ie., diseases)
    structures (ie., chemical)
advice (ie., experiment plans)
service (ie., circuit layout)
analysis (ie., mineral exploration)

2.6 Construction of expert systems (knowledge engineering)

The development of expert systems has brought into existence the new field of knowledge engineering, whose major focus is upon acquiring, representing and utilizing the knowledge human beings apply when solving problems (Fei79). At present, this discipline is very much a collection of ad-hoc methods (and, as such, is a suitable domain for another expert system (like AGE)).

There are four major areas of activity associated with the development process. These are overviewed here and discussed in greater detail in sections 5.3 and 5.8. They are:

model formation - the developer provides for representing and using the domain-specific knowledge. The first step is to develop a domain model (a difficult problem). The next step is to decide how the knowledge is to be encoded, so that it conforms to the constraints of both the model and the computing device. In conjunction with this, one has also to decide upon the computational elements of the system: the type, manner and order of its reasoning.

knowledge acquisition - given the subject area, the developer must somehow acquire the knowledge that is to be used in the expert system. The first task is to find a set of human experts (or references) who have the desired knowledge. The next task, in that the knowledge is often not in an explicit form, is to get the expert to articulate and record their expertise in the context of the model being used for the domain (see next area). Clearly, the knowledge acquisition process requires much interaction between the knowledge engineers and experts, and typically takes many months of work.

knowledge debugging - since the domain-specific knowledge is initially induced by human experts, and is then encoded by human knowledge engineers, errors, inconsistencies and omissions in the knowledge base are inevitable. Correcting the knowledge base requires the combined efforts of the experts, knowledge engineers, and the system itself (describing its reasoning processes).

system tuning - after the system works correctly (ie without logical errors), its logical performance has to be optimized. This requires not only adjustment of parameters, but also iterative improvement of the model and the knowledge base.

All of these activities relate to the representation and use of the domain specific knowledge. There are also the system engineering tasks, consisting of machine and user interface device selection, design and development of the underlying expert system reasoning program, development of the user interface and user support features (such as explanation generation) and general performance (ie., speed) enhancement. These activities can require considerably greater effort than the knowledge engineering tasks. For this reason, the use of a domain-independent system (ie., EMYCIN, HEARSAY-III), which provides knowledge engineering tools, should be investigated when developing a new expert system.

2.7 Some actual implementation details

This section summarizes some performance data obtained from working expert systems. Note that the development time in these cases (except SACON) includes the time needed to develop the underlying expert system framework. The SACON system, which was built on the EMYCIN domain-independent expert system framework, required mainly the knowledge acquisition and debugging tasks.

SYSTEM	MACHINE	PROGRAM SIZE (K WORDS)	NUMBER OF RULES (DOMAIN)	ESTIMATED RESPONSE TIME (SEC)	ESTIMATED DEVELOPMENT TIME (MAN-MONTHS) (ENGINEER/EXPERT)	ESTIMATED SESSION TIME (MINUTES)
MYCIN	PDP 10	245+	450+	???	???	20
R1	PDP 10	206	666	150	12/3	???
SACON	PDP 10	???	170	???	???/4	25

Table 2 - a summary of expert system production details

2.8 A critique of expert system research

This section discusses some criticisms of expert systems from the point of view of their suitability for use in tutorial systems. A majority of these points are discussed by Clancey (Cla79c) in his PhD thesis on the **GUIDON** system.

The major criticisms center about the domain knowledge, its source and use:

1. Besides containing valid rules, the knowledge base must be sufficient and complete in the items and methods covered. (By sufficient, we mean that the knowledge contains enough detail to support humanly intelligible reasoning, and by complete we mean that it has enough knowledge to correctly reason about all anticipated problems.) However, Clancey points out that, because of the optimized form of the knowledge, many of the rules are nonsensical out of context. Also, the data bases are often formed by combining the efforts of many rule authors, which can lead to inconsistency. For example, **MYCIN**'s rule base consists of two separate groups of rules, one concerned with clinical data based reasoning, and the other with laboratory data based reasoning. In general, the heuristic nature of the knowledge, coupled with the individual rule-author styles, interferes with the tutorial goals of clarity and consistency.

2. The nature of the optimized rule form omits all information regarding the actual interpretation of raw data, the meaning of trends and the underlying mechanism whereby the rule arises. As a result, the rules acquire a superficial character, in that the reasoning depends upon use of a variety of heuristics that have little relation to the actual process. This also means that the problem solving methods are often weak in some aspects (**EMYCIN** does not handle time relationships well, which are of importance when considering disease as a process).

3. There is a major deficiency in the type of reasoning done by the systems. In general, reasoning steps are fairly superficial (with the more precise domains, such as gamma-ray physics, the reasoning becomes more precise). The model of inquiry is also weak as well; human experts have some sense of what conclusions to look for at particular times, whereas most expert systems have a more random approach to the solution. (This also varies with the precision of the domain and model). What is needed is some meta-level knowledge, which dictates which sub-problems are to be solved first and how.

4. Clancey also points out difficulties that expert systems will have in tutorial applications. Again, reasoning is a major factor, in that the system's style of reasoning is usually not similar to the type desired for the human, nor does it generally reflect any knowledge of the underlying processes in the domain. The lack of meta-level knowledge also hinders tutoring, in that the system has no real concept of key factors or key principles to address when facing a problem.

5. Another problem is that the expert system generally depends upon a single model of the domain, whereas a human expert may make use of several, depending upon the scale and stage of problem solving. This lack limits the effectiveness of the mechanical problem solving, as well as the appropriateness of explanations about the system's reasoning.

6. Lastly, the whole procedure used by the expert system may be unsuitable for tuition, as in the GAMMA system. While the overall structure of the solution procedure and domain model is worth knowing, the major value of the system comes from its ability to rapidly search its voluminous tables of physical data, and its ability to heuristically test a number of alternative solutions, optimizing quickly to a solution.

3. Educational Systems Background Discussion

This section provides a background discussion of the research in intelligent teaching systems, especially those which require an underlying expert performance program. It covers both general research topics and specific systems.

3.1 General overview

In the early years of computer-aided instruction (CAI), work focussed upon methods of information presentation. A curriculum was decided upon by a human teacher, who then developed a series of fixed presentations on the chosen topic. These presentations were transcribed into computer-readable format. The student was passively subjected to the presentations, much as if at a lecture. Clearly, in this form, CAI differed little from linear programmed learning, with everything pre-determined by the designer. Later work allowed for some branching between alternative treatments according to the results of simple comprehension tests, but was basically concerned with handling deviations from the norm and did not individualize learning (How73).

The major deficiencies in this approach include:

1. no manipulable model of the domain, so it was unable to handle open queries from the student

2. restricted communication - no adequate language handling capability (keyword input only)

3. weak (statistical) models of the student, and no inference of what he/she learns or ought to be capable

4. fixed teaching strategy

As a result of the limitations of existing CAI systems, investigators began to consider what was required in a sophisticated CAI system. One set of requirements is Hartley's framework for adaptive CAI (Har73), which is summarized in the four points below. A CAI program needs to have:

1. knowledge of the subject to be taught

2. knowledge of the student, and his/her performance

3. knowledge of suitable teaching methods, and

4. a theory of how to apply such skills in particular cases

Papers by Clancey, Bennett and Cohen (Cla81b), Gable and Page (Gab80), Howe (How78) and Self (Sel77) discusses the above issues and others in greater detail.

3.2 Problems addressed by intelligent teaching systems

In this section, we expose some of the tough issues which must be tackled within the context of the requirements drawn by Hartley:

1. Knowledge of the subject

   a) curriculum structure - the problem is to establish the correct static relationship between the concepts that are to be taught. Elements of both a sequential nature (concept B logically follows concept A) and a hierarchical nature (concept B is a generalization of concept A) have to be accommodated.

   b) concept selection - given a range of concepts that are currently suitable for tuition, the problem is to decide which is best suited as the next topic for presentation.

2. Knowledge of the student

   a) intuitive vs. formal knowledge - the problem is whether (and how) the student's existing intuitive knowledge should be represented as a subset of the expert's formal knowledge. One current technique is the "overlay model" (Car77), in which a student's knowledge is presumed to be a literal subset of that known to the teacher.

   b) error diagnosis - the problem is to deduce the student's factual or procedural errors, based upon performance errors. This requires a much more sophisticated student model, in that the system needs to know not only the correct methods, but also a majority of the possible erroneous methods. Further, a method is needed to deduce which error or errors produce the observed behavior.

   c) explanation understanding - the problem is to infer a student's reasoning processes and knowledge of facts from his/her explanations or answers to questions.

3. Teaching Methods

   a) The principle issue is when to use a particular teaching method. Methods available include explanation, exercising, experimentation, and socratic dialog. The selection of methods depends upon particular teaching theories.

4. Teaching theory

   a) active vs. passive - the issue here is the choice of teaching philosophy, which is determined by personal beliefs about how people learn. Those who advocate passive methods favor explication and exercise, whereas those who believe in an active approach favor experimental and socratic methods.

   b) intervention - in the context of active methods, the problem is to know when the tutor should let the student take a weak action without comment or interrupt with advice. This difficulty occurs in realistic problem solving domains, because a surplus of trivial corrections can distract the student from perceiving the major

issues. The intervention decision depends on whether the advice is appropriate to the student's current knowledge level, the current tutoring goals, and good psychology (ie., have we interrupted the student too much lately?).

c) Interaction management - the issue under consideration is the maintenance of the dialogue between the student and tutor, the details of which include: who chooses the next topic, how much information is to be presented, how to present the topic, how long to stay on the topic, who asks and who answers questions, when to consider a topic satisfactorily discussed and how to keep to the topic.

3.3 Examples of intelligent teaching systems

In the early 1970's, research focussed on the representation of the subject matter, the first of Hartley's rules. But as Brown (Bro77) pointed out, "ICAI systems cannot be AI systems warmed over", so research has widened in scope to include the student model and teaching strategies. Example systems include which explore one or more of the issues outlined in section 3.2 are:

BIP (Bar76) - a tutor for the BASIC programming language. It is based upon a (humanly) generated structure of skills and techniques organized into a curriculum. Associated with each task in the curriculum is a programming exercise, which has the student writing a simple computer program. The underlying system attempts to evaluate the student's computer program automatically. Task selection is based upon a model of the student's weak and acquired skills. The system also offers hints and demonstration programs.

WUMPUS/WUSOR II (Gol78,Car77) - a system built around a maze exploration game. The function of this game is to teach students facility with the use of deterministic and probabilistic logical reasoning in the context of a game situation. The expert for playing this game has its knowledge codified as a small set (about 25) of production rules. It attempts to induce which rules are applied by the player and then to teach new rules (in appropriate situations) according to a rule syllabus. Student models are formed according to estimated knowledge of which rules are known to the student. (The term "overlay model" is used in this situation because the student's known rules overlay the set of rules known to the system.) Attention is paid to teaching rule generalizations and detecting when the student appears to have learned a new rule. Research also covers effectiveness of the student's representational techniques (ie., maps, lists, arrays, etc) in the game context.

FLOW tutor (Gen77) - a tutor for the FLOW programming language. Here the tutorial method is organized into a set of schemas, each covering specific techniques. The schemas are organized into a hierarchy of concepts. Of interest is that the system keeps an agenda of schemas to be taught in the near future. Each schema has an associated specialist program responsible for the contents of that schema. The tutorial role of this system is to track the student's progress through a workbook, and to provide advice and criticism when necessary.

ACE (Sle77,Sle79) - a program which analyzes a student's reasoning through his/her interpretation of an NMR spectra. The student can enter the explanations in somewhat natural English, but the argument as a whole has to conform to specific argument forms. The system translates the input into a formal logical language and then evaluates the statements against its knowledge of the chemical structure used for generating the particular spectra. The system tolerates some missing facts used implicitly in the reasoning.

BLOCKS (Bro78a) - an attribute guessing game based upon information given about which objects fit/do not fit the attributes. (It is a bit like the popular MASTERMIND game). The goal is to teach procedural and reasoning knowledge. Key notions are those of contradictory and sufficient evidence for a hypothesis. The program also attempts to teach some efficiency notions and some abstractions from the actual

objects. The student model is kept according to the rules that the system believes the student knows.

BUGGY (Bro78b) - a system for automatically modelling and diagnosing children's subtraction errors. The research assumed that the student's errors largely arose from correctly following incorrect procedures (ie., misconceptions). They modelled the overall subtraction method as a procedural network (a collection of low level procedures organized to perform a complete task), with errors being introduced by way of incorrect variants in subprocedures. BUGGY was designed to give student teachers training in diagnosing student subtraction problems, but has also been used to actually diagnose the conceptual errors underlying the incorrect behavior.

WEST (Bur79a) - a system built around a stagecoach board race game. The function of the game is to teach students facility with arithmetic expressions. It has an underlying expert which makes optimal moves. On top of this is a "coach" program, which induces whether the student has addressed certain "issues" (in both arithmetic and the strategy of the game) in the course of selecting his/her move. The program builds a model of the student's knowledge of these issues, based upon analysis of the moves actually made. The coach program then chooses appropriate times to interject comments and suggestions of better moves, thus exposing the student to other arithmetic principles.

QUADRATIC TUTOR (OSh79) - a tutor for the quadratic formula (solves second degree polynomial equations). Of interest here is that O'Shea embeds a tutorial strategy into a set of production rules, which then reason over both the student model and the program material.

3.4 Intelligent teaching systems built upon expert systems

These systems can be distinguished from other intelligent teaching systems by being built upon expert system bases. The systems, **SOPHIE**, **EXAMINER**, and **GUIDON**, are discussed below:

1. **SOPHIE** is a system designed to teach students how to trace faults in an electronic apparatus. It operates by presenting the student with a circuit schematic of the device, a stabilized low-voltage power supply, into which it has previously introduced a fault of some specific degree of difficulty. The student's task is to trace this fault. After requesting various circuit measurements, the student suggests what he/she thinks might be wrong and request replacement of various devices. Then, the program checks that the solution is correct and is consistent with what was learned about the circuit from making the measurements. **SOPHIE** can also give hints regarding hypothesized failures consistent with the measurements known to the student. The student communicates to **SOPHIE** in a reasonably natural English dialogue (which can also resolve anaphoric references, by virtue of the restricted domain of discourse).

SOPHIE uses powerful inference-making procedures to evaluate the student's solution to the problem in the particular context determined by the set of measurements he/she has made. The key feature of these inference methods is the incorporation of an analysis package for simulating analog circuits. Hence, **SOPHIE** is capable of simulating the power supply circuit, using the underlying simulation expert. For example, if the student suggests that the fault is due to the failure of a particular transistor in a feedback circuit, by running the simulator with this component failure, a repeat set of measurements is obtained. These hypothetical measurements are compared with those taken by the student, and, if disagreement is found, the student is informed of the discrepancies. Notice that the program does more than merely match the values; it needs to know about values in a working circuit to identify measurements that support the proposed solution and those that are independent of it.

SOPHIE is a very large program. In fact, it occupies some 300k words of memory on a DECsystem-10 machine. Despite its complexity, the investigators claim that the average response time to a question is three seconds. While this is a satisfactory measure of the program's efficiency, we lack evidence about its educational value. One built-in assumption is that the student will have some knowledge of the basic electronic principles underlying the design and operation of DC power supplies. The measurements the student makes take on meaning in the context of his/her mental model of the power supply, and this enables the student to put forward a hypothesis. But, if the mental model is wrong, the solution will probably be wrong too. Unfortunately, **SOPHIE** is not sufficiently intelligent to explain why it is wrong. Asking **SOPHIE** for help may not be of any help either, since the alternative hypotheses suggested by the program also assume the student's mental model is a sound one.

2. **EXAMINER** (Ole77) addresses another tutorial goal – that of testing the student's knowledge. It is built upon the INTERNIST (Pop77) medical diagnosis expert system, and uses its diagnoses and medical knowledge to evaluate the student's responses.

A session with **EXAMINER** is intended to resemble an oral examination taken by a medical student. The student is presented with the facts of a

selected case (which come from either a case library or are entered by the student). The student then enters his/her diagnosis, which consists of listing the major independent disease processes, with linkages to complications or related diseases. He/she also has to list the symptoms for which this diagnosis is to account. **EXAMINER** then compares this information with the diagnosis of the case as generated by **INTERNIST**. The results of this comparison are then returned in 3 forms:

1. **EXAMINER** lists the major (bad) facts not accounted for by the student's diagnosis, and presents additional diagnoses.
2. Disease categories are refined if the case facts are sufficient to give a more precise diagnosis
3. Alternative interpretations not considered by the student are suggested.

EXAMINER fulfills a role similar to that of **SOPHIE** - the student can test his/her diagnostic abilities against a number of selected cases. Unlike **SOPHIE**, though, the student has no "experimental" capabilities. **EXAMINER** makes up for this by having a more sophisticated criticism capability. Another major feature of **EXAMINER** is it allows the student to enter the diagnoses at any acceptable level of generalization, and hence is appropriate for students at various levels of development. A further feature is the ability to accept multiple disease diagnoses, with alternative hypotheses - a characteristic of real medical diagnosis.

On the other hand, **EXAMINER** is not as sophisticated as a human examiner. There is no focus upon the student's reasoning - that is, no attempt is made to diagnose the reasoning errors made by the student. A further problem is that the "examination" is not genuinely interactive. The dialogue format is literally: fact presentation, collection of student's diagnosis, critique of diagnosis. A session with a human examiner would allow for more specific detailed questioning by both the student and examiners. Another minor point is that all of the case facts are presented immediately to the student, which is unlike real cases or oral examinations, in which the student (or doctor) would have to deduce which new facts should be acquired based upon the current hypothesis. A final problem concerns the criticism of the student's reasoning, which is strictly factual (ie., is based about the diagnosis paradigm) rather than centered about the underlying medical processes. The criticism only tells the student what's wrong with the diagnosis, rather than why he/she should have known it was wrong.

3. The **GUIDON** system is the most interesting system that we will discuss, in that it consists of a intelligent tutorial system built directly upon an existing expert system, the famous **MYCIN** system.

GUIDON, built by Clancey (Cla79a, Cla79b, Cla79c) represents a unified attempt to address many of the tutorial issues discussed in section 3.2, as researched in the systems discussed in section 3.3. These include curriculum control, student modelling, dialogue management, opportunistic tutoring and evaluation of responses. In some sense, it is an expert system itself, whose expertise is in understanding **MYCIN**'s reasoning, and communicating it to the student. The intent of the tutoring is to make the student aware of gaps and inconsistencies in his/her reasoning as he/she uses the basic domain knowledge. This presumes a reasonable initial level of development on the part of the student.

The interaction between the student and the system is oriented about the discussion of (student) selected cases from **MYCIN**'s repertoire -

medical diagnosis cases covering meningitis and bacteremia. The intent is for the student to diagnose the case in the same manner as **MYCIN**, by being given the same initial knowledge and then attempting to duplicate **MYCIN**'s logic (that is, substantially recreate MYCIN's AND/OR solution tree). This examination is largely directed by **GUIDON**, which suggests avenues of inquiry to follow next, provides hints to help the student's inquiries and deductions, and summarizes large sections of simple reasoning, based upon assumptions of the student's current abilities. This examination is focussed about and limited to the solution generated by **MYCIN**. (In fact, the student only learns about deductions that can be made given the data currently known, much as a doctor would. As the student requests more case data, as if interviewing a patient, more of the solution tree is made accessible to **GUIDON**, which then directs the student in making further deductions.)

The **GUIDON** system is actually constructed to work with any expert system built using the **EMYCIN** domain-independent structure. Clancey, in his PhD thesis (Cla79c), gave example sessions also using the **PUFF** and **SACON** expert system rule bases. While **GUIDON** did succeed in working with these rule bases, it was felt that the SACON example was a failure. This was largely due to the fact that the type of solution generated by SACON was technical, very directed and relied upon full knowledge of the problem parameters, whereas, in the **MYCIN** case, the result depends upon deducing only a few items, but with a variety of possible methods and little initial data. The result was that the **SACON** tutorial was extremely verbose, had little interaction with the student and presented most of the results directly. (Clancey argues that this may be remedied by an alteration to the tutorial strategy of **GUIDON**.) It was further noted that the **MYCIN** rule base had to be augmented slightly to provide all of the necessary facilities for **GUIDON**. (This mainly consisted of adding schemas and supporting text to the original rules.)

We now look a several segments of dialogs with **GUIDON** copied from figures 7 and 8 of a paper by Clancey (Cla79a). (Reproduced as figures 4 and 5 here). In the examples, the student's inputs are in the boldface type and comments by Clancey are in paired brackets ("[","]").

You should now proceed to ask questions that will enable you to make a hypothesis about the organisms (other than those seen on cultures or smears) which might be causing the infection.

{Initiative is given to the student; he may ask for more data; make a hypothesis; or use a special option.}

** HOW IS THE FACT THAT THE PATIENT IS BURNED RELEVANT?

{One option is to ask how a piece of data is relevant to the current goal.}

The fact that patient-326 is seriously burned will be used to conclude that *Pseudomonas aeruginosa* is one of the organisms which might be present (in addition to those seen on cultures or smears) [Rule 578].

** HELP

{The "help" option is a request for assistance: the student asks "where do I go from here?"}

Try to determine the type of the infection: bacterial, fungal, viral, or TB.

{To provide help, GUIDON chose: (1) a rule to discuss (D-rule 578 was chosen because the dialogue just focused on it); and (2) a method of presentation—"tell the student to finish incomplete subgoals mentioned in the rule." Thus, a new topic is set up, "the type of the infection." Initiative is returned to the student . . .}

** WHAT IS THE PATIENT'S WBC?

The white count from the patient's peripheral CBC is 1·9 thousand.

{The tutor observes that this question is relevant to the new topic and provides the requested data. The dialogue continues in this goal-directed manner . . .}

FIG. 7. Sample interaction: gathering data.

Figure 4 - GUIDON dialog with student gathering problem data

{After presentation of relevant case data . . .}

** I THINK THAT THE ORGANISMS PRESENT ARE DIPLOCOCCUS-PNEUMONIAE, PSEUDOMONAS-AERUGINOSA, AND NEISSERIA-MENINGITIDIS.

Good. The age of the patient gives evidence for concluding that Diplococcus-Pneumoniae might be present (Rule 507). The fact that the patient has been seriously burned is good reason to suspect *Pseudomonas aeruginosa* (Rule 578). Both conclusions apply because the infection which requires therapy is meningitis, no organisms were seem on a gram stain of the culture, and the type of the infection is bacterial.

The age of the patient also leads one to suspect *Neisseria meningitidis* (Rule 507). Do the fact that the infection was acquired while the patient was hospitalized and information about the diseases to which the patient has been exposed support this conclusion? ** NO

Right, this would be good reason to rule out *Neisseria meningitidis* (Rule 561).

What facts about this case tell you that the organisms which might be present are: *E. coli* and *Klebsiella pneumoniae*?
** HELP
{The student was expected to describe each factor with a phrase, one per line. Help is provided by listing all relevant factors for the current goal.}
Choose from:
1. The patient does not have a TB risk factor and the patient's x-ray is not abnormal
2. Otitis-media and Epigolottitis are not among the diagnoses which are consistent with the patient s clinical history
3. The white count from the patient s peripheral CBC is less than 2·5 thousand
4. The infection was acquired while the patient was hospitalized
5. The patient has been seriously burned
6. None of these

** 1 3 4 {This question was a form of "entrapment." Notice that the choices all appear in Fig. 9.}

You're right about the WBC (Rule 557) and the evidence gained from the fact that the infection was acquired while the patient was hospitalized (Rule 545). However, the lack of a TB risk factor and a normal x-ray is weak evidence that Mycobacterium-TB is not one of the organisms which might be present (Rule 160).

FIG. 8. Sample interaction: hypothesis evaluation.

Figure 5 – GUIDON dialogue with system evaluating student's hypothesis

Figure 4 illustrates the student finding out more information about the patient. At roughly line 17 the student asks for help. **GUIDON** responds by directing the discussion towards the discovery of a particular subgoal, the type of infection, at which point the student starts to make inquiries towards this end. In figure 5, we see **GUIDON** analyzing the student's hypothesis on the organisms responsible for the infection. When the student's answers do not agree with MYCIN's, **GUIDON** takes the opportunity to quiz the student (line 12). Further, it attempts to query the student on several other potential organisms (lines 19/20) which leads to the student receiving help in the form of a multiple choice quiz and discussion (lines 25-43).

It is noted by the author that these dialogues illustrate **GUIDON** at its best.

We now discuss some of the important facets of **GUIDON**.

tutorial strategy - **GUIDON** bases its expertise on a set of about 50 packets of tutorial rules (about 200 rules total). Each packet is responsible for a particular tutorial action. The tutorial rules (t-rules) are distinct from the domain rules (d-rules) of the underlying expert system. The t-rules are responsible for the selection of the next system response, the evaluation of the student's inputs, the updating of the student model and the general control of the dialogue.

The major educational strategy used is that of opportunistic tutoring, that is, to correct the student or to present new material when opportunities arise during the normal course of the dialogue. These opportunities occur when the student displays unexpected expertise, when the student needs a new rule to make a conclusion and when new case data gives rise to new conclusions. The decisions about when to tutor and what to say are also embedded in the t-rules.

Two sample t-rules are shown in figure 6. The first rule, **T-RULE 2.04**, chooses a method for discussing a d-rule. In this case, it considers a rule which could be used except for one missing premise fact. The rule leads the discussion to considering that missing fact (which involves other t-rules), and then summarizes the results of the current d-rule. The second rule, **T-RULE 7.01**, is used to help update the belief that a particular rule was used by the student in the course of reasoning out a conclusion.

T-RULE 2.04

IF: (1) The number of factors appearing in the d-rule which need to be asked by the student is zero, and
(2) The number of subgoals remaining to be determined before the d-rule can be applied is equal to 1

THEN: Substep I. Say: subgoal-suggestion
Substep II. Discuss the goal with the student in a goal-directed mode [Proc001]
Substep III. Wrap up the discussion of the rule being considered [Proc017]

T-RULE 7.01

IF: You believe that the domain rule was considered, and
It concludes a value present in the student's hypothesis, and
No other rule that mentions this value is believed to have been considered

THEN: Modify the cumulative belief that this rule was considered by 0.40

Figure 6 - two GUIDON tutorial rules

dialogue management - GUIDON attempts to control the complete dialogue. This is done by focusing the dialogue about a particular case, and generally discussing it in a manner of its choice. The sequence of the dialogue goes as: discuss student's comments, give general discussion, ask student a question. At each of the question points, the student is free to continue the current discussion or to proceed in his/her own direction (with the particular case remaining the general area of discourse). The system attempts to keep the student on the current topic. The overall structure of the dialogue, as well as the types of transitions allowable during various segments of interaction, are all specified, and easily described in a state transition diagram. The transition format was constructed on the basis of the patterns present in a normal tutorial discourse, and summarizes many of the strategies that a human tutor would use while discussing a problem with a student.

Beside the research issue of how to present the information, there remains the question of what to present. The full MYCIN case solution consists of a considerably greater amount of information than is reasonable to present in a one hour tutorial. As a result, the system attempts to limit its discussion. Some methods for this are: summarizing reasoning that the student is believed to already know, stating trivial conclusions and only stating key factors in a deduction. (The automatic decision of what is a key factor is a significant research problem. GUIDON uses the approach of only stating those factors which don't fit associated rule schemas - which are presumed to be prototypical rules or situations.)

Input interface - The input interface is mainly command word driven. Commands include: IKNOW, HYPOTHESIS, DISCUSS, QUIZ, HINT, TELLME, SUMMARIZE and about 20 others. MYCIN's pattern-matching keyword parser is used to analyze input, and thus allows some simple natural language inputs.

student modelling - The major technique used is the overlay model. In GUIDON, it is presumed that the student's knowledge is a subset of the rules used by MYCIN. GUIDON attempts to assign belief that the student knows each rule. (Belief is a numerical factor in the range -1 (certainly doesn't know) to +1 (certainly does know)). Evidence for belief is based upon background information (general student development level), explicit evidence (from quizes) and implicit evidence (deductions from student hypothesis and conclusions).

Overall, Clancey (Cla79c) gives the following list of major design principles intended in the construction of GUIDON:

perspicuousity
assume errors imply missing rules
strictly guide the dialogue
orient discussion by top-down refinement
assist by small advances
opportunistic quizing and extensions
quiz to verify models

We now conclude with some criticisms of GUIDON. Most of these points are discussed by Clancey in his PhD thesis (Cla79c).

The first point is about the use of the domain knowledge itself. The knowledge is summarized in the d-rules which EMYCIN uses to reason about the problem. The major failing of this approach is that the rules

capture what could be considered as a "black-box" view of the solution process, that is, they describe the inputs and outputs of the steps of the solution process, but not the reasoning itself. Hence, if the student learns the rule, the deduction made would be valid, but he/she would have no understanding of why. Some of this information loss comes from the expert's not having it in the first place, and secondly from the rule structure being optimized to remove intermediate conclusions subsumed by the overall conclusion. (This relates to the "grain size" of the rules, that is, the level of detail that the domain model captures.) There is also a problem of the rules being somewhat nonsensical out of context of particular problems. Lastly, the rules often contain numerical ranges for symptoms and certainty factors for conclusions that do not correspond to human notions (such as "high fever") and are not easily learnable.

The next point is with the student modelling. Obviously, the student's knowledge will be structured in some way other than the system's (production rules). Further, the knowledge is likely to be other than just a subset of the rules: rather the two knowledge bases are likely to share some common subset of knowledge. In his thesis, Clancey discusses these issues, nowever, all of his examples show only static knowledge of the domain, and at general beginner levels. Hence, it isn't clear whether the modelling is effective, or if the model dynamics are accurate. Clancey also points out that the modelling presumes that the student remembers everything learned.

The system does not attempt to diagnose the student's errors, rather it is just content to report them. This stems partly from the lack of a deep model of the student's reasoning.

A more significant problem lies with the use of **GUIDON** with other knowledge bases. Clancey's thesis has examples of its use with the **PUFF** and **SACON** knowledge bases. The **PUFF** tutorial was successful; the **SACON** one was not. SACON's reasoning structure is detailed, precise and consists mainly of sequentially determined decisions. This contrasts with **MYCIN**, whose decisions are few and depend upon weighing considerable alternative evidence. As a result, the SACON dialogue is more of a monologue. Further, the reasoning summarized is a bit detailed and analytic. Clancey suggests that this result relates to the AND/OR solution tree structures: **MYCIN**'s is flat and bushy, whereas SACON's is tall and sparse.

Some student comments on the **GUIDON/MYCIN** system were that it was overly verbose and that its selection of tutorial methods was so varied that it gave an unsettling feel of unpredictability.

Clancey comments that a major source of **GUIDON**'s limitations stem from a lack of any strategic knowledge, that is:

1. rules describing how to apply the domain rules,
2. ways to summarize methods and rules,
3. methods to comprehend and complete the student's reasoning
4. lesson planning.

4. Training System Alternatives

This section contains a speculative discussion of what training might an intelligent system perform and how, what sort of interaction would it have with a student and the feasibility of constructing such systems.

4.1 What roles might potential expert training systems have?

To answer this question, we have to give some thought to the type of user. Most probably, these systems should not be used by naive students, in that even the simplest forms of an expert's reasoning are likely to be confusing (if not incomprehensible) to a student unfamiliar with the basic concepts, relations and structure of a subject and its problem solving techniques. Next, we have to consider whether the users are knowledgeable but casual (ie., once a year users), or are fully involved (as students or as active, but developing, users). This depends upon the role of the system. We have delineated five potential roles for an expert training system and four potential structures in which to embed the roles.

The roles are:

Adviser - the system provides advice to the user who is attempting to solve a problem. This is a fairly minimal training system, in that the training comes by way of the experience gained in the course of solving actual problems. The burden upon the system is substantial, in this case, in that the range of potential problems or user's needs (ie., areas of uncertainty or lack of knowledge) is wide.

Demonstrator - this system demonstrates the solution of typical problems, in order to give the student some conception of the use of domain methods. The student's role is relatively passive, but he/she could influence the direction and level of detail in the discussion. This role is probably easiest to implement, however, for best performance, the tutorial system requires knowledge not ordinarily possessed by the underlying expert system. This is the meta-level knowledge used to direct problem solving, which is then used to motivate the discussion of the solution (ie., why particular actions were taken).

Examiner - the system evaluates the student's problem solving or decision making capabilities. This probably entails the training system presenting the student with a problem, soliciting a response, and critiquing the student's solution. The system has to recognize the structure of this solution, determine how it relates to its own solution, and abstract the principles behind the differences. A further capability would be inducing the causes behind the student's failures. Unless the structure of the examination is rigidly controlled, this type of system is considerably more advanced than current research.

Exerciser - the focus of this system is upon exercising the user's existing skills - so as to help him/her gain experience with actual problem solving and understanding of the relevance of factual knowledge. This type of system has varying development requirements, depending upon the form that the exercise takes and the degree of domain-specific understanding that the system needs.