AI Magazine Volume 8 Number 3 (1987) (© AAAI)

A Graduate-Level Expert Systems Course

David C. Brown

This article presents an approach to a graduate-level course in expert, knowledge-based, problem-solving systems. The core of the course, and this article, is a set of questions called a profile, that can be used to characterize and compare each system studied.

It seems that these days everyone wants to take a course in expert systems! The problem is how to teach it in order to make it anything more than a look at a few applications of a few AI techniques.

The Computer Science Department at Worcester Polytechnic Institute offers a course called Expert, Knowledge-Based, Problem-Solving Systems to graduate students who have already taken the graduate-level introduction to AI. The course size is limited to 20. It is run as a 14-week course, with one 3-hour class per week.

One goal of the course is to examine a number of expert, knowledge-based, problem-solving systems, looking at each system in some depth. Another important goal is to make comparisons across systems in a domain-independent way. An attempt is made to relate systems by their problem-solving capabilities rather than merely by the AI techniques used.

Each student is set the goal of understanding at least one system in detail. The student becomes the expert on, and the representative of, the system. The goal of improving the student's presentation skills is satisfied by having each student make a presentation about a particular system. Another goal is to have each member of the class contribute during class discussions.

Course Content

The content of the course as currently taught is given in table 1. The references are merely pointers to the literature and are not intended to be complete.

The first two lectures are used to organize the course, present a general introduction to expert systems, and discuss some advanced topics. Most systems are presented by the students, with a final lecture by me about my research. The choice of systems is intended to provide a view of some of the most well-known, successful, and influential systems, yet it allows for a contrast between different types of problems and different approaches to their solution. The systems also cover a wide range of claims about how much they resemble human problem solving.

There is some disagreement about which systems are in fact expert systems. For example, some lists include speech-understanding systems. Although humans could be called expert speakers, I don't think that we would call them experts in the speech domain. I have tried to only include systems where the domain is such that it is learned as an adult and where we would be inclined to acknowledge that someone could be called an expert.

Teaching Method

At the start of the term, the students are given a package of readings that provide background material for the whole course as well as specific material for the first two lectures. This material includes an introduction to how the course is to be run and a description of its goals. It also includes surveys, comments on the field, and technical papers.

The assignment of students to systems is done during the first meeting. Wherever possible, it is done according to student's requests because some students have appropriate domain knowledge, and others (part-time students) find some link with

Example Profile, MDX/PATREC

Domain. Medicine; especially a subset of liver diseases: CHOLESTASIS

Main General Function. Input and organization of a patient's medical data (PATREC), and diagnoses based on these data to determine the cause(s) of cholestasis

System Name. Diagnostic component: MDX (Medical Diagnostic Expert); database component: PATREC (Patient Record)

Dates. Approximately 1978-1983; some ongoing development

Researchers. B Chandrasekaran, F Gomez, S Mittal, J Smith, J Sticklen, T Bylander

Locations. Laboratory for Artificial Intelligence Research, Ohio State University, Columbus, Ohio

Language. MDX is written in LISP; PATREC is written in LISP and FRL An implementation language (CSRL) for developing MDX-type expert systems is also available

Machine. MDX was initially implemented on a DEC-10, PATREC on a DEC-20

Brief Summary. This system consists of two major components: MDX, the diagnostic component, and PATREC, the database component Each component is responsible for all knowledge and reasoning in its own area For both MDX and PATREC, knowledge is distributed throughout a hierarchy of major concepts relevant to either diagnosis or medical data This hierarchy is called the conceptual structure In MDX, the concepts represent diseases or disease categories The diagnostic task is also distributed among these concepts Each one solves a small part of the diagnostic problem by establishing itself (if possible) as a partial explanation of the symptoms, and then by refining itself with calls to its sub-concepts to achieve a more specific diagnosis for the symptoms At many points during the diagnosis, MDX queries PATREC for information PATREC answers by following links in its own hierarchical organization of patient medical data to retrieve specific values or make inferences from them as necessary

Related Systems. RED is an expert system used for red blood cell antibody identification in blood banks Its architecture and distributed approach to problem solving are based on MDX, but RED has an abductive assembler that produces a single best identification MDX-MYCIN is a version of MDX written for a sub-domain of MYCIN (meningitis), and it has been used to compare the performance of MYCIN's global uncertainty calculus to MDX's local approach AUTO-MECH is a small expert system that tests the MDX approach in a non-medical domain, and also examines the usefulness of CSRL for implementing expert systems of the MDX type

Characterization of Givens. MDX contains knowledge about the logical relations between its classificatory concepts (partial hypotheses) Each concept contains local knowledge about conditions under which it may establish itself and how to refine this hypothesis by calling sub-concepts PATREC contains knowledge about the logical relations between its data frames and their attribute slots (inheritance, inference, etc) PATREC also contains some anatomical knowledge, temporal reasoning logic, and flexible input parsing knowledge

Characterization of Output. MDX currently produces as output a list of possible diseases, each of which may account for a particular set of cholestasis-related symptoms The list is accompanied by the set of symptoms that support each disease PATREC produces two types of output: specific values and trends (such as white blood cell count), and logical responses to queries, which can have values true, false, or unknown

Characterization of Data. Input data is via PATREC, and consists of a set of medical data pertinent to the diagnosis of cholestasis (clinical findings, lab tests, medical history, etc), organized by major temporal episodes (e g "at admission") These data may be incomplete or unreliable in a variety of ways: (a) test results may be lacking; (b) test results may be of questionable validity; (c) the actual relationship between symptoms and the disease(s) which cause them may be unclear

Generic Tasks. PATREC uses knowledge-directed information passing; MDX's main task is classification, but hypothesis matching and abductive assembly play a major role in the classification process

Theoretical Commitment and Reality. Three major claims are made about the theoretical underpinnings of MDX The first is that its conceptual structure represents one valid model of how experts view knowledge in the domain, and that the hierarchical nature of this structure promotes focused problem solving Several sources of information (textbooks on diagnosis, consulting human experts) support this claim The second claim is that a modular, local uncertainty calculus is more realistic than a global one, since the same piece of knowledge may have different meaning and/or diagnostic value in different contexts MDX is most definitely an attempt to simulate the method of human diagnosticians, rather than simply to reproduce their diagnoses A final claim is that symbolic uncertainty measures represent a logically valid method for combining components of evidence, a method which closely resembles the reasoning of the experts themselves

Completeness. MDX currently performs diagnoses in the sub-class of liver diseases characterized by cholestasis, and within this subclass it is fairly complete The possibility of implementing diagnoses of other liver diseases exists In the current implementation, MDX produces a list of likely diseases but does not choose the "best" diagnosis from among them Work has been done on an abductive assembler that could perform such a task One such assembler has been implemented for RED (see "related systems")

Use and Performance. The system has not been tested in an actual clinical or diagnostic setting It has been run on test cases drawn from the medical literature, and it produced reasonable diagnoses in these cases The performance of MDX's local uncertainty calculus was measured in comparison to MYCIN's global calculus and was in general found to be comparable (although it performed less well than MYCIN when both were denied significant laboratory findings about the patient's condition)

Phases. The first phase of operation is data entry via PATREC The patient's symptoms, history, and so forth are entered and organized in PATREC's patient record Initial inferences regarding the data may then be drawn Search is then initiated in MDX, starting with the root node Successor nodes are called in a recursive establish/refine process in order to form a list of plausible diseases with supporting evidence Finally, this list is checked to see if all the abnormal data are explained

Subfunctions. MDX does not recommend therapy or perform other sub-functions outside of diagnosis

Use of Simulation or Analysis. No numerical simulation or analysis is performed

System/Ctonrol Implementation Architecture. MDX is based on a hierarchy of specialists (active agents), each of which organizes a set of rules In principle these active agents could be processed in parallel and communicate via a blackboard The current implementation is serial

Characterization of the Structure Knowledge. Knowledge in MDX is grouped into a hierarchy of concepts that represent disease categories at various levels of abstraction Each concept has local, highly specific knowledge about how the concept establishes itself and how it refines its successor concepts Both the diagnostic knowledge base and the inference engine are distributed among these concepts A similar approach is taken in PATREC, where patient medical data is organized into a hierarchy of frames with attached inference procedures

Characterization of Process Knowledge. The main effect of a concept's knowledge is to establish whether or not that concept is a relevant component of the final diagnosis Thus, active knowledge serves either to confirm or to exclude certain concepts (hypotheses), and to recommend which sub-concepts should be tried next The net result is a classification: <subset of symptoms> —> <partial explanation> The partial explanation would include evidence for or against a given concept

Deep or Surface?. The knowledge in MDX is highly compiled, surface knowledge The claim is made, however, that the organization of knowledge into a conceptual structure captures the result of all "relevant" deep knowledge, and that compiled knowledge is therefore adequate for all diagnostic tasks within the domain

Search Space. The search space in PATREC is composed of all possible paths through the MEDATA hierarchy from general data category to very specific data values The MDX search space consists of all possible paths through the cholestasis hierarchy from general disease category to very specific hypotheses about the disease The search states are explicit in the hierarchical knowledge representation Search states represent successive refinements of a diagnosis

Space Traversal. In MDX, after a concept establishes itself, it uses local knowledge to refine itself: its children are called to try to establish themselves, and this establish/refine process continues until the diagnosis cannot be pursued further If a concept cannot establish itself, all its successors are ruled out, resulting in considerable pruning of the search space In PATREC, a global query processor analyzes queries and directs search to one or more patient record frames Local query knowledge associated with each frame may infer values by inheritance through the procedural triggering of related searches, or sometimes from default values

Search Control Strategy. The control strategy is called Establish-Refine This strategy may be expressed as follows: Consider a general disease category that at least partially accounts for the symptoms Then consider successively more specific diseases as causes, accounting for more of the symptoms, until the most specific cause(s) are reached This represents the collective diagnosis (in medical jargon, the "differential")

Standard Search Strategies. MDX can be considered to use a form of the generate-and-test search strategy, except that the available choices are always given in the concept hierarchy The abductive assembly process implemented in the related RED medical expert system uses an explicit form of means-end analysis

Search Control Characterization. Top-down, best first search driven by local knowledge In some cases, an established concept might consider all its children if information about which is most likely is not present In this case, the best-first search degenerates to depth first

Subproblems. Concepts represent distributed problem solvers Each concept forms a partial solution to the overall diagnostic problem Concepts establish or reject themselves, so each "knows" if it is on the right track or not Sub-concepts are largely independent of their siblings, but not of their parents This would not necessarily be true of any expanded implementation designed to deal with multiple disease hierarchies

Search Control Representation. Search control representation is largely explicit Some control knowledge is explicit in the conceptual hierarchies of MDX and PATREC Some control knowledge is explicit in a concept's local production rules or in a frame's procedural attachments Other control knowledge in PATREC's global data acquisition and query modules is more implicit

Search Control Strength. Methods are strong: they are domain dependent (except for some of PATREC's global query processing module), and knowledge rich

Failure Method. PATREC rejects input data for which it cannot find an appropriate frame or slot; it also responds "unknown" to a query if the data are insufficient to infer an answer "Unknown" must be clearly distinguished from "false" or default values In MDX, local knowledge is used to establish or reject individual concepts (hypotheses) If data exclude a hypothesis at some level of confidence, that concept is rejected along with all its successors, resulting in a significant pruning effect If the data are insufficient to establish or reject a hypothesis, the concept suspends itself If a complete line of reasoning fails, control returns to the highest established concept along this line, which then tries other possibilities If a normal search cannot account for all the data, an exhaustive search through all likely possibilities may be tried

Uncertainty. Both data and knowledge may be uncertain, and as a result solutions to subproblems may also be uncertain

Management of Uncertainty. MDX uses a fixed range of 7 certainty values, represented by the integer range -3 (very unlikely) to +3 (very likely) These represent an ordered set of likelihoods; no numerical calculations are performed A local method of combining certainty values is used, and each concept uses its own specific situation-dependent knowledge-based approach Certainty values are combined using expert specified tables, that associate a resulting certainty with each meaningful combination of the component values Not all combinations are represented However, it is likely that many of those not represented at a given node may have been ruled out in prior processing

Management of Time. Much of the patient data is time-dependent: when certain symptoms began, what their trend has been, etc The actual temporal dependencies are inferred by PATREC, which clusters temporal data into "episodes" such as at-admission and at-surgery PATREC then orders these episodes as linearly as possible Imprecision of time estimates is also considered: three weeks may be represented as ((years 0)(weeks 3)(days unknown)), since 3 weeks might be anything from 18 to 24 days

Knowledge Representation Method. Mainly hierarchically organized clusters of local rules, with some look-up tables, and with local procedural knowledge to handle message passing between concepts

Knowledge Representation Generality. Although MDX was written in LISP, there is now a special language, CSRL, for building classification systems CSRL facilitates the construction of a hierarchy of local specialists whose knowledge is represented in one or more "Knowledge Groups" (KG) A KG contains a set of rules or a table One group is devoted to summarizing the other groups conclusions

Knowledge Structuring. Knowledge is organized as a hierarchy of concepts, each of which contains specific, local knowledge about itself This hierarchy is called the conceptual structure The claim is made that the hierarchy of concepts should be similar or identical to the way an expert would organize his knowledge of the domain This hierarchy helps to focus the problem solving process

Alternative Structuring. The system doesn't currently use alternative representations for data

Alternative Solution Methods. None

Optimization and Multiple Results. MDX produces a list of diseases, each of which accounts for some subset of symptoms with an explicit degree of certainty To get the best diagnosis from this list of candidates, some sort of abductive assembly is required MDX currently lacks this capability, although it has been implemented in RED (see "related systems")

Interaction. While the system is being developed, if any concepts are invoked which have not yet been implemented in detail, an interactive human expert can be "plugged in" to play the role of the missing concept MDX and PATREC also use some interesting acquisition interactions (see below)

Data Collection and Acquisition. Typically all patient data are made available to PATREC at the outset of a diagnosis These data are made available to MDX on an as-needed basis, and MDX can query the user for relevant data that are lacking in PATREC PATREC guides the user in data acquisition by prompting for categories of data (e g lab tests), and by requesting that all data from the same temporal episode be entered at the same time The input module prompts for data in an order compatible with the hierarchical organization of the patient data base

Data Format. Data are input in the form of a patient record The actual input uses a flexible LISP-like syntax For example, the datum "patient exhibits mild pain in the epigastrum" can be input as (PAIN EPIGASTRUM MILD) or as (PAIN MILD EPIGASTRUM)

Learning. None

Explanation. When a concept is established, it maintains a list of the symptoms that supported it The final diagnosis contains, for each plausible disease, a list of all the symptoms that supported all the concepts that led to the inclusion of the disease The system can also supply a trace of its goals during execution Because of the organization inherent in the conceptual structure, this trace can give the appearance of a causal explanation

Strengths. Conceptual structure has a strong intuitive appeal as a valid model of how experts organize domain knowledge. The hierarchical nature of this structure allows focused problem solving with significant pruning It permits local control of reasoning and local uncertainty handling It facilitates modular implementation and local enhancement, and it can also lend itself readily to distributed processing techniques

Weaknesses. Distributed knowledge and control structures can be costly to implement without tools such as CSRL, and, even with such tools, global modifications may be difficult MDX's use of compiled tables for hypothesis matching or for combining uncertainties does not facilitate in-depth justifications of the inference process Systems which rely on hierarchical representation methods may be forced to incorporate complex control mechanisms in order to handle some realistic problems such as multiple diseases

Other. This is quite enough!

This example profile was written by Bill Sloan and Doug Green for CS 525, Topics in Computer Science: Expert, Knowledge-Based, Problem-Solving Systems

Week	Topic	Speaker	Reference(s)
1	Organization, Introduction, and so on	Brown	Hayes-Roth 1984 Gevarter 1982 Feigenbaum 1979 Marr 1981 Davis 1982
2	Generic tasks, deep vs. surface	Brown	Chandrasekaran 1986 Hart 1982 Chandrasekaran and Mittal 1983a
3	MDX/PATREC/CSRL	Student(s)	Chandrasekaran and Mittal 1983b
4	MYCIN/EMYCIN	Student(s)	Buchanan and Shortliffe 1984
5	TEIRESIAS	Student(s)	Davis and Lenat 1982
6	INTERNIST/CADUCEUS	Student(s)	Pople 1982
7	CASNET-glaucoma/EXPERT	Student(s)	Kulikowski and Weiss 1982
8	PROSPECTOR	Student(s)	Duda and Reboh 1984
9	AM/EURISKO	Student(s)	Davis and Lenat 1982 Lenat and Brown 1983
10	MOLGEN	Student(s)	Stefik 1981 Friedland 1979
11	DENDRAL/ Meta-DENDRAL	Student(s)	Lindsay et al. 1980
12	SU-X/HASP/SIAP	Student(s)	Nii et al. 1982
13	R1/XCON	Student(s)	McDermott 1982
14	AIR-CYL/DSPL	Brown	Brown and Chandrasekaran 1986

Table 1. Content of the Expert, Knowledge-Based, Problem Solving Systems Course.

their work. Verbal ability is an important factor in the course because every student has to make a presentation. Because it is often the case that some of the students are not native speakers, every attempt is made to pair these individuals with a competent speaker. Each three-hour presentation is made by one or two students.

After each presentation, the presenter(s) provide(s) a written profile of the system. This completed profile is copied and then distributed at the start of the following class. The outline of the profile (see the next section) is provided, thus ensuring a consistent format for the summaries. The concepts used in the profile make it essential that every student has already had a course in AI.

The week before each presentation, a paper about this system is distributed to the class as a reading assignment. This preparation is essential if the class is to fully absorb the details of each system. In addition, it raises the quality of the in-class discussions. Preparation can be tested and, to a certain extent, enforced by directly asking students for their comments.

As an additional test of the students' ability to evaluate systems, a final profile about my research is required from each class member. This report is produced only from the readings and is collected at the start of the lecture in which the AIR-CYL/DSPL system is presented. All other profiles are done with the benefit of having heard class discussion as well as the instructor's comments and questions.

Because much of the literature about expert, knowledge-based, problem-solving systems is hard to find, a collection of major papers is provided for each presenting person or group. Two weeks in advance of their presentation, the students are given a folder containing about 5 to 10 papers relating to their system. By limiting the access to material in this way, an attempt is made to equalize preparation time across the class.

Using a Profile

After the presentation, the specific person or group is expected to provide the class with a completed profile of the system. The purpose of providing this profile is to delineate a substantial set of attributes by which systems can be categorized and subsequently compared. This profile is also intended to guide the form of each presentation to the class, although this structure does not need to be strictly adhered to. The profile was developed with the help of sources such as Gevarter (1982) and Hayes-Roth, Waterman, and Lenat (1983). However, it primarily comes from discussions, seminars, and courses in which I have been involved (see "Acknowledgments")

For several systems, some profile questions are easy to answer; for other systems, they are hard. Despite my best intentions, some of the questions overlap. Because the class discussion contributes to the understanding of each system, I think it is better that the profile be completed after the class in which the system is presented The completed profile is submitted to the instructor prior to the next class so that copies can be made, to be distributed the following week. The collection of completed profiles is a very important benefit of this class.

The following section contains the format for the profiles the students are to provide. A sample profile of the MDX/PATREC system, taken from the course, follows at the end of this article.

The Profile

The profile is divided into specific sections. Each section has a number of headings that represent questions to be asked The questions in each section are related in some way Comments or sample answers follow the headings.

Domain. Chemistry, electronics, computers, and medicine (plus specific subarea)

Main General Function. Diagnosis, prediction, planning, and data interpretation

System Name. AM, DENDRAL, and MYCIN

Dates. Rough period of development

Researchers. Main people responsible for research and development (R&D)

Location. Where R&D took place (can be more than one)

Main References. The clearest, as well as the most comprehensive

Language. Lisp, OPS5, KEE, Loops, and ART (can be more than one)

Machine. Lisp Machine, DEC20, and VAX

Brief Summary. A short paragraph describing the system

Related Systems. Systems from which this system was developed, systems developed from this system, and systems by others using same methods (clones)

Characterization of Givens. What is the information given and built into the system (expressed as abstractly as possible)?

Characterization of Output. What is the information produced by the system (expressed as abstractly as possible)?

Characterization of Data. Is the data reliable? Is the data complete?

Generic Tasks. Which generic tasks are obviously included, explicitly or implicitly, for example, classification, state abstraction, knowledge-directed information passing, object synthesis, hypothesis matching, abductive assembly, and simple selection?

Theoretical Commitment. Does the system have any theoretical underpinning? Is it claiming to show that some theory of its type of problem solving is correct? Is the method used claimed to work for other similar domains?

Reality. Is there any psychological validity to the method used—the structure of the knowledge, the control mechanisms? Is it a system that is merely a simulation of result, or is it in any way a simulation of

Completeness. Has the system been fully implemented? Has all the domain been included?

Use. Has the system been used with real users from outside the original development situation? Has the system been used with real users in the user's own working environment?

Performance. Are there any performance measures available? How was the system evaluated? How did it fare?

Phases. Is the system organized into distinct phases of different activity? Distinct subtasks? What are they?

Subfunctions. Despite the fact that the system has a single main function (for example, diagnosis), does it use other types of problem solving as part of the system (see the answers to the last question for clues to whether this answer might be yes)? What kind(s)? For example, diagnostic systems sometimes select therapy, too; this can be done by synthesis or selection (which is often a kind of classification; that is, can this disease situation be classified as one which this drug can help?).

Use of Simulation or Analysis. Does the system use a numeric simulation or analysis, either done by itself or some package, during its operation?

System-Control Implementation Architecture. What is the overall architecture? For example, metarules + rules, blackboard + knowledge sources, production rules, active agents, activation nets, and so on.

Characterization of the Structure Knowledge. Is it grouped into types? What types, for example, component knowledge, chemical knowledge, functional knowledge, and causal knowledge? What are the types used for? Do they correspond to phases of the system?

Characterization of the Process Knowledge. For active knowledge, how would you characterize the effect of this knowledge, for example, <partial situation description>—> <classification> ?

Deep or Surface. Is the system using deep knowledge, or is it only using surface knowledge? How would you characterize it? Are there levels of representation or reasoning? For example, if there is a qualitative simulation, it could be argued that it is using deep knowledge.

Search Space. What space or spaces does the system search through? Is the search explicit? Are the states represented explicitly? How? What do the states represent, for example, complete alternative solutions, solution refinements, and plans? How big is the space?

Space Traversal. How is the space traversed? What does it mean in terms of the problem to move from one part of the space to another, for example, subproblem decomposition, gradual refinement, moving down a predetermined hierarchy, and instantiation?

Search-Control Strategy. Does the system use a strategy that is expressible in terms of the problem, or does it appear just to be an AI technique that happens to fit?

Standard Search Strategies. Does the system explicitly use generate and test (G&T) or means-ends analysis (Be careful, almost everything is a form of G&T!)? How would you characterize the search control, for example, depth first, best first, breadth first, knowledge based, and random?

Subproblems. Is evaluation of partial solutions possible (that is, can the system know when it is on the right track?)? Are the subproblems independent? Totally? Partially?

Search-Control Representation. Where is the search-control knowledge? How is it expressed? Explicitly?

Search-Control Strength. Is it based on a very domain independent and knowledge-free method (a weak method), or is it very domain dependent and knowledge full (a strong method)?

Failure Method. When part of the system fails or reaches an incorrect conclusion, how does the system attempt recovery? What type of knowledge does it use? Is it global or local? Is there a significant variation from the normal flow of control? If it uses backtracking, what kind does it use?

Uncertainty. What is it that is uncertain? A piece of data? A piece of knowledge? A solution to a subproblem? If all are used, how do they interact?

Management of Uncertainty. Does the system use probabilities, scoring values, a fixed range of certainty values? What does a value mean? What is the method of combina-

tion? Is it a global method, or does it vary (that is, a local method)? Are there any apparent problems?

Management of Time. Is there any time-dependent data? How does it affect the problem solving?

Knowledge Representation Method. What method(s) does it use to represent knowledge? For example, rules, procedures, tables, semantic nets, and logic.

Knowledge Representation Generality. Is there a special language for this system, or does it use a general method? Is the method provided by some expert system building tool?

Knowledge Structuring. Is the system based on a hierarchy, a network, or some other structure? Does this structure correspond to the domain in some way? Does it correspond to the problem solving being carried out?

Alternative Representations. Does the system use alternative representations for the same piece of knowledge in order to allow for alternative solution methods?

Alternative Solution Methods. Does the system use alternative methods to reach the same solution(s)?

Optimization. Does the system produce the best answer? Sometimes? Always? How? Why does it need to?

Multiple Results. Does the system produce more than one result? If it obtains several, does it try to order them by evaluating them? Does it try to combine them? If so, how?

Interaction. Is there anything interesting about the way the system interacts with the user (for both input and output), for example, by using menus, pictures, diagrams, picking, color, or sound?

Data Collection. Does the system require all its data (that is, details of this particular problem) before execution, or does it allow or require incremental addition? If incremental, does it vary the type, number, or order of data-gathering actions (for example, questions of a database or user) depending on the problem being solved?

Data Format. In what form is the data given, for example, patient records, Lisp, or natural language?

Acquisition. Does the system have any way of acquiring knowledge from the expert user? Does it guide the user? Does it have a way of validating the knowledge?

Learning. Does the system learn from its own performance? If so, how? What does it learn?

Explanation. Does the system have the ability to explain where its result came from? Is this obtained from a trace of the goals and subgoals formed during system execution? If not, how is it formed?

Strengths. What do you think are the strong points of the system?

Weaknesses. What do you think are the weak points of the system?

Other. Any other comments that help characterize this system.

Conclusions

The students and I have found the course to be very successful. However, it is not without problems.

One major criticism is that not every student has the same amount of work to do. Those working by themselves have to prepare a three-hour presentation, but others only prepare a one-and-a-half-hour presentation. This situation can be compensated for somewhat by insisting that each member of a group submit a separate profile. Another factor, however, is that some systems are harder to understand than others, perhaps because of the complexity of the system or the quality of a researcher's writing. Systems that are perceived to be more difficult than others can be assigned a larger group.

Because there are no exams, some students feel at the start of the class that the course doesn't entail much work. However, one student complained that he spent 45 hours preparing his talk and profile. Because this amount works out to only three hours per week on the average, it actually seems quite reasonable.

Because different people have different presentation skills, some talks are much harder to understand than others. The reading distributed the week before helps to ensure that everyone understands the material. In addition, the instructor should summarize the major points every so often during each presentation. Discussion can be prompted by asking students for summaries. In addition, comparison between systems is facilitated by calling on one of the student representatives of some other system.

If the presentation is poor as a result of language problems (that is, a nonnative speaker), the instructor needs to play an active role. However, presentations can be improved by insisting that the students prepare overheads of major points and diagrams prior to the class.

A problem that might occur in the future is students cheating by using profiles produced for a previous course. Because previously submitted profiles are not included in the folders of technical papers, the students would have to seek out the prior presenters of a system. Cheating would require the cooperation of another advanced graduate student, which is possible but not likely. My view is that students get out of a course an amount proportional to what they put into it. If they cheat in the course, they cheat themselves, too. However, one way to combat the problem is to gradually change the list of systems to be examined.

The course places some burden on the instructor. Of course, the instructor should be able to answer all the profile questions for every system—if not before each presentation, at least afterward! The only weekly work is to read and grade the profiles submitted, do the reading prior to class, and work out some key questions (the moral) in advance. The major effort occurs before the first class when you identify and collect all the major papers on the chosen systems. Save a whole drawer of your filing cabinet!

The benefit obtained by the students depends highly on the quality of discussion, which in part depends on the quality of the students. However,

the instructor must play an active part in stimulating and controlling discussion and must be prepared.

Apart from the few problems mentioned, the comments from students on their teaching evaluation forms have been extremely positive. The presentation approach using a profile is an excellent way to provide an in-depth comparison of expert systems for a graduate-level course.

Acknowledgments

The author would like to acknowledge the intellectual stimulation that has been provided by members of the Laboratory for AI Research at Ohio State University and members of the AI Research Group at Worcester Polytechnic Institute I would also like to acknowledge Drs. Doug Green and Bill Sloan for allowing me to use their profile of MDX as an example

References

Brown, D C., and Chandrasekaran, B 1986. Knowledge and Control for a Mechanical Design Expert System *IEEE Computer* 19(7):92-100

Buchanan, B , and Shortliffe, E H 1984 *Rule Based Expert Systems* Reading, Mass : Addison-Wesley

Chandrasekaran, B 1986 Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design *IEEE Expert* 1(3):23-30

Chandrasekaran, B., and Mittal, S 1983a Deep versus Compiled Knowledge Approaches to Diagnostic Problem-Solving. *International Journal of Man-Machine Studies* 19:425-436

Chandrasekaran, B , and Mittal, S. 1983b Conceptual Representation of Medical Knowledge for Diagnosis by Computer: MDX and Related Systems. In *Advances in Computers*, vol 22, ed M. Yovits, 217-293. New York: Academic.

Davis, R 1982. Expert Systems: Where Are We? And Where Do We Go From Here? *AI Magazine* 3(2):3-22

Davis, R , and Lenat, D 1982 *Knowledge-Based Systems in AI* New York: Academic.

Duda, R O , and Reboh, R. 1984 AI and Decision Making: The PROSPECTOR Experience In *AI Applications for Business*, ed. W. Reitman, 111-147 Norwood, NJ : Ablex

Feigenbaum, E A 1979. Themes and Case Studies of Knowledge Engineering In *Expert Systems in the Micro-Electronic Age*, ed D. Michie, 3-25. Edinburgh: Edinburgh University Press.

Friedland, P 1979. Knowledge-Based Experiment Design in Molecular Genetics. Ph.D diss., TR-CS-79-771, Dept. of Computer Science, Stanford Univ

Gevarter, W B 1982 An Overview of Expert Systems, Technical Report, NBSIR 82-2505, National Bureau of Standards

Hart, P E. 1982. Directions for AI in the Eighties *SIGART Newsletter* 79:11-16

Hayes-Roth, F 1984 Knowledge-Based Expert Systems *IEEE Computer* 17(10): 263-273

Hayes-Roth, F ; Waterman, D. A.; and Lenat, D B. 1983 *Building Expert Systems* Reading, Mass : Addison-Wesley

Kulikowski, C., and Weiss, S 1982 Representation of Expert Knowledge for Consultation: The CASNET and EXPERT Projects In *AI in Medicine*, ed P Szolovitz, 21-55. Boulder, Colo : Westview

Lenat, D., and Brown, J S 1983 Why AM and Eurisko Appear to Work In Proceedings of the Second National Conference on Artificial Intelligence, 236 Menlo Park, Calif.: American Association for Artificial Intelligence.

Lindsay, R.; Buchanan, B ; Feigenbaum, E. A ; and Lederberg, E 1980 *Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project* New York: McGraw-Hill.

Marr, D. 1981 AI: A Personal View In *Mind Design*, ed. J. Haugeland, 129-142 Cambridge, Mass : MIT Press.

McDermott, J 1982 R1: A Rule-Based Configurer of Computer Systems *Artificial Intelligence Journal* 19(1):39-88.

Nii, H. P; Feigenbaum, E A ; Anton, J J.; and Rockmore, A J 1982 Signal-to-Symbol Transformation: HASP/SIAP Case Study *AI Magazine* 3(2):23-35

Pople, H 1982. Heuristic Methods for Imposing Structure on Ill-Structured Problems: The Structuring of Medical Diagnosis In *AI in Medicine*, ed. P Szolovitz, 119-190 Boulder, Colo : Westview

Stefik, M 1981 MOLGEN *Artificial Intelligence Journal* 16(2): 111-169.