ARTIFICIAL INTELLIGENCE APPLICATIONS TO COMPUTER-ASSISTED INSTRUCTION

Project Progress Report No. 1:

BASIC CONCEPTS IN KNOWLEDGE-BASED SYSTEMS

Jesse M. Heines, Ed.D.
Staff Consultant

Systems Based Courseware
Educational Services Development & Publishing

April 20, 1982

Educational Services
Crosby Drive
Bedford, MA 01730

1.0 ABSTRACT

The purposes of this research are to investigate computer-assisted instruction (CAI) courseware development tools that have the potential to improve

- the quality of CAI courseware and/or
- the efficiency of the CAI course development process.

For the past few months I have been pursuing these goals by investigating the applicability of artificial intelligence (AI) concepts to CAI. While several researchers have worked in this precise area, little working software exists to implement their ideas. I have therefore been studying AI concepts through more generally available software packages, and one of these is described extensively in this report to demonstrate the bases and power of AI.

The report contains the following major sections:

- The Domain of AI -- a brief, general overview of three major classes of AI systems
- Concepts in Knowledge-Based Systems -- an introduction to the basic concepts of one class of AI systems
- The AL/X System -- a detailed discussion of an AI system that we have running in Educational Services, including a description of how the knowledge-base is structured, how the driver program functions, and how the system looks to a user
- A Conceptual Look at a Rule-Based Tutor -- a description of one possible instructional application for knowledge-based systems being developed by the British Open University
- A Partial List of AI Efforts at DEC -- synopses of some of the other AI efforts going on in the Corporation
- AI/CAI Research Efforts to be Pursued -- a look at where we might go from here

As these section headings show, my research is still very much in the conceptual and "information gathering" stage, so I would appreciate input from readers on additional avenues that we might profitably pursue.

2.0 TABLE OF CONTENTS

3.0 THE DOMAIN OF AI

Artificial intelligence is a large and amorphous subject field, typical of a field in its infancy. To make matters worse, it is "in" these days to say that one is working on AI, so many projects seem to be described in AI terms when they are only loosely related to basic AI research.

Basic AI research falls into three major domains:

- image processing and pattern recognition,
- natural language parsing, and
- knowledge-based systems.

3.1 Image Processing and Pattern Recognition

Image processing refers to a variety of techniques for manipulating images digitized with a computer. At the low end, such techniques might include some of the manipulations that can be done with Genigraphics, or even our own DRAW program. The difference, however, is that while the source data for Genigraphics and DRAW are commands, the source data for AI image processing systems are usually actual images.

Pattern recognition is a form of image processing, but for a very specific purpose: to identify recognizable patterns in the image and thus "decipher" it. At the low end, pattern recognition might include the reading of Universal Product Codes that appear as bars on most grocery products. At the high end, this application is an important component of robotics, where robots are built to perform tasks based on visual input. For example, robots exist today that can perform different tasks depending upon differences in visual input that they discern using pattern recognition. Note that the SPEEX terminal (which responds to voice input) also uses pattern recognition techniques, but the patterns that it recognizes are aural rather than visual [1].

3.2 Natural Language Parsing

Natural language parsing is concerned with making computers able to understand native human language. While some work in this area has been conducted in the United States and Britain, it appears that there is considerably more interest in other countries. This phenomenon may be due in part to the fact that most other languages are more structured than English (possessing full conjugations of verbs and declensions of nouns) and are therefore

---

[1] For more information on SPEEX, contact Stew Kranz in Educational Services Development & Publishing.

considerably easier to parse.

Natural language parsers can be valuable for analysis of student responses in a CAI environment. In CAI parlance, such analysis is usually referred to as "answer judging". Readers interested in this particular area should look at the PLATO answer judging techniques developed mainly through the efforts of Paul Tenczar at the University of Illinois. These techniques are an integral part of PLATO's Tutor language [2].

3.3 **Knowledge-Based Systems**

Knowledge-based systems are software systems that attempt to "mimic" the deductive or inductive reasoning of a subject matter expert. (Such systems are also known as "expert" or "rule-based" systems.) The distinctive characteristic of these systems is that their logic processes are state-driven rather than hard-coded. That is, one "programs" a knowledge-based system by writing a set of logic rules that the system digests during initialization. Program branching is then controlled by these rules rather than by IF/THEN structures hard-coded in the source program.

Knowledge-based systems are particularly suitable for applications such as medical and mechanical diagnosis. In fact, the most famous knowledge-based system is one known as MYCIN (Stanford University) which identifies bacterial diseases from a large number of characteristics of cultures grown from sample bacteria. The next two sections of this report discuss the concepts involved in such systems and demonstrate these concepts through a program called AL/X, a Pascal "mini-version" of MYCIN that we have running on a VAX system in Educational Services.

---

[2] Ken Moreau in ES D&P's Computer-Based Course Development group has experience with these PLATO facilities, and some of these capabilities have been implemented in Dimension, the Tutor-like CAI authoring language being marketed by the Education computer Systems group.

4.0 CONCEPTS IN KNOWLEDGE-BASED SYSTEMS

4.1 Rules

The logic in most computer programs is coded through one of a variety of computer languages. This logic may direct the computer to access data stored in a disk file, but the decisions about how to process that data are almost invariably made by "hard-coded" logic in the computer program. In knowledge-based systems (and most other AI systems), the program logic itself is stored in a database in the form of rules. (Such a database is often referred to as a knowledge-base, hence the name knowledge-based systems.) These rules direct the computer to perform certain actions depending upon which rule is applicable for the current state of the program.

AI rules, when considered as a set, generally form a hierarchy of condition/action pairs known as an associative or inference network. This network can be thought of as the core image of a knowledge-base, ready to be accessed by an AI driver program. Such inference networks differ from simpler constructs such as binary trees in that the rules may be parents of several rules and children of several other rules. The lines connecting nodes in an inference network often cross each other, where this would be invalid in most simpler network constructs.

The major part of AI programming is the loading, interpretation, and use of the inference network by a driver program. Rules establish "hooks" to each other as they are loaded, rather than by a predefined structure. This characteristic yields a fascinating property of AI inference networks: the rules that comprise them can be loaded in any order. That is, if a knowledge base consists of five rules, the inference network would be the same if these rules were loaded in the order 12345, 54321, or 24135.

A corollary of this property is that an inference network can be extended simply by adding new rules to the knowledge base, and these rules may be added at any location in the knowledge base. Thus, as more information is learned about an AI problem, the AI program's expertise can be enhance simply by adding new rules. Neither the driver program nor the original structure of the knowledge based has to be altered.

4.2 Instantiation

Once a knowledge-base is loaded and an inference network has been established, an initial state of the network is defined. The AI driver program then begins to collect additional data about the users' problem, usually through an interactive dialogue with the user at a computer terminal. As each new piece of information is gathered, the driver analyzes that information to determine which

rules apply.

Whenever the conditions for a rule match the state of the inference network, the action part of that rule is exercised. This sequence is known as instantiation. That is, specific states of the network cause instantiation of certain rules.

The key to the AI process, however, is that each such instantiation changes the state of the inference network, and thus affects the instantiation of other rules in the future and the ultimate decisions that will be made by the AI program. You might think of rule instantiation as a chain reaction, or ripple effect, in which the state of the inference network triggers instantiation of a certain rule, which in turn changes the state of the inference network, which in turn triggers instantiation of another rule, etc.

After the driver program has fully digested a piece of information entered by the user and propogated the effects of this information throughout the inference network, the program either makes a positive or negative decision regarding the hypothesis that it is investigating or asks the user for additional information. This cycle is repeated until a decision can be made or the user has entered all the information at his or her disposal.

4.3 Boolean vs. Bayesian Decisions

Boolean concepts are the bases of most standard computer programs. They are based on repetitive use of the construct:

IF A THEN B

Boolean concepts are the conceptual bases for simple, concrete decisions. They are highly applicable to decisions involving discrete data and situations in which there is usually only one "correct" answer. These are standard computer operations, and most computers have specific hardware components to speed up Boolean comparisons and decisions.

Bayesian concepts, on the other hand, are based on the construct

IF A (to degree x) THEN B (to degree y)

These concepts form the conceptual bases for complex, abstract decisions. They are applicable to decisions involving phenomenological data and situations in which there might be more than one "correct" answer. These are non-standard computer operations, and generally require extensive software to support them.

The difference between Boolean and Bayesian concepts can perhaps best be understood with an example. Suppose the decision you wish to make is whether you should take your umbrella when you

leave your house or apartment on a specific day. The Boolean approach to this decision might consist of two simple (and redundant) rules:

(a) If it is actively raining when you leave, take your umbrella.
(b) If it is not actively raining, don't.

The Bayesian approach would be more complex, involving a number of factors that weight the decision one way or the other. Assume that you begin evaluating the decision with a score of 0. Factors that indicate you should take your umbrella add positive values to the score. Factors that indicate you should not take your umbrella add negative values to the score. In the final analysis, if the score ends up positive you will take your umbrella, and if it ends up negative you will not. Further assume that the values to be added to the score vary between -100 and +100.

Factors that might add positive values to the score are:

(a)	+80	It is actively raining when you leave.
(b)	+50	The weatherman says it is going to rain today.
(c)	+30	Your spouse says it is going to rain today.
(d)	+20	You must wait outside for a bus to get to work.
(e)	+20	The sky is overcast.
(f)	+15	You are wearing new clothes.
(g)	+15	Some meetings today that will take you outside.
(h)	+5	You have just bought a new umbrella.

Factors that might add negative values to the score are:

(i)	-50	It is not actively raining when you leave.
(j)	-40	The weatherman says it will not rain today.
(k)	-25	Your spouse says it is not going to rain today.
(l)	-25	You can drive yourself to work today.
(m)	-20	The sky is clear.
(n)	-15	You are wearing old clothes.
(o)	-15	You know that you will be staying inside today.
(p)	-15	You have lots of things to carry into work today.

Note that related positive and negative factors are not weighted merely as opposites. The weightings shown above bias the decision toward taking the umbrella if the factors tend to cancel each other out. Such bias can be built into AI systems if desired when it is known a priori that the cost or risk of one decision (leaving the umbrella home and getting wet) is greater than the cost or risk of another decision (taking the umbrella when it is not needed).

Virtually all AI systems operate with Bayesian rather than Boolean concepts. This is due to the nature of the decisions that most AI systems try to make: decisions in which circum-

stantial rather than concrete evidence exists, and in which decisions must be based on probabilities rather than totally black and white situations.

5.0 THE AL/X SYSTEM

Educational Services has been experimenting with a knowledge-based system called AL/X which is written in Pascal. We negotiated a one-year license for this software (through the efforts of Stew Kranz) and had it installed on a VAX system. AL/X allowed us to "get our feet wet" quickly, and we found it both an excellent tool for introducing ourselves to knowledge-based systems and a highly satisfactory system for demonstrating basic AI concepts. The program authors describe the software as follows (Paterson, 1981).

> AL/X (Advice Language/X) is a program which allows the encoding of the rules used by an expert to solve a problem and provides an inference engine [a driver program -- JMH] to use this knowledge to give advice to the user. An important characteristic of [this] expert system is its ability to explain its reasoning.
>
> Much of this work is based upon the Prospector consultant system for mineral exploration developed at SRI International by Richard Duda, Peter Hart, et al. AL/X has been developed for Intelligent Terminals Ltd. [the company that currently markets it -- JMH] by John Reiter, Steve Barth, and Andy Paterson. This work was done in association with the University of Edinburgh [Scotland] and was supported by British Petroleum Development Ltd.
>
> The aim of the initial project was to develop an expert system to diagnose the underlying causes of a certain category of oil platform shutdowns. AL/X has since been used to develop expert systems for other diagnostic problems.

AL/X has two distinct components, a knowledge-base and a driver program. The knowledge-base is a file containing the rules that will be used to solve the user's problem. These rules, when processed by the AL/X program, form an inference network of "evidence/hypothesis" relationships. The AL/X program itself acts as both a "knowledge acceptor" during construction of the inference network and as an interactive consultant to generate and explain advice.

5.1 The AL/X Knowledge-Base

The source for generating an AL/X knowledge-base is a sequential text file that contains a number of coded hypotheses or assertions called spaces. These hypotheses have associated with them numerical degrees of belief and are related to each other by links of various types.

Each of these three constructs is discussed conceptually below, but the discussion stops short of describing how these constructs are coded in the knowledge-base. A partial example of knowledge-base coding is provided with the sample application in Section 4.3.5. Readers interested in further coding details should contact Jesse Heines for a copy of the AL/X User Manual.

5.1.1 Spaces

Spaces are the major building blocks of knowledge-bases and fall into three categories.

5.1.1.1 **Questions** - represent information bearing on the problem the system is trying to solve. Such questions pertain to problem details that the system expects the user to be able to provide. A distinguishing feature of most question spaces is that they have no antecedents. All question spaces have consequents (or there would be no point in asking them). Questions, then, are entry points into the inference network.

5.1.1.2 **Intermediate hypotheses** - represent partial conclusions that the system might draw from the user's responses to questions. Such hypotheses do not solve the user's problem, but they do allow the system to delineate the problem by establishing some parameters with a relatively high degree of certainty. Intermediate hypotheses have both antecedents and consequents and are thus central points in the network.

5.1.1.3 **Goals** - represent possible solutions to the user's problem. They are the highest level of the system's decision-making capability. A goal is established -- either positively or negatively -- when the driver program has tested all rules (by asking all relevant questions) that bear on that goal. The system may then go on to investigate the evidence supporting other goals, but the current line of reasoning will be terminated. Goals always have antecedents, and generally they have no consequents. Thus, goals are exit points from the network.

5.1.2 Degrees of Belief

Each space in the knowledge-base has associated with it a numerical degree of belief. This number represents the probability that its associated hypotheses is true.

A prior degree of belief for each space is specified during coding of the knowledge-base. This "prior" degree actually represents a bias as previously discussed. It indicates the *a priori* probability that a space is true before any additional

information is collected from the user.

Degrees of belief are the major "currency" of a knowledge-based system. As the user enters information in response to questions, the system changes the degrees of belief for all consequent spaces. Hypotheses and goals are termed "true" and "false" when these degrees reach prespecified threshhold levels.

In AL/X, the value of a degree of belief may vary between -100 and +100. A value of 0 indicates that the likeliness of a hypothesis being true is equal to the likeliness of it being false. In other words, the probability that a hypothesis with a degree of 0 is true is 50%. Positive degrees indicate increasing probability that a hypothesis is true, while negative degrees indicate decreasing probability.

AL/X interprets degrees of belief using an "odds/log" formula. This formula may be expressed as

$$\text{degree}(H) = 10 \times \log_{10}\left(\frac{\text{prob}(H)}{1 - \text{prob}(H)}\right)$$

Table 1 shows the mathematical relationships between degrees of belief, odds, and probabilities for selected values.

Table 1

RELATIONSHIPS BETWEEN DEGREES OF BELIEF,
ODDS, AND PROBABILITY

(after Paterson, 1981)

Degree of Belief	Odds	Approximate Probability
-30	1:1000	.001
-20	1:100	.01
-10	1:10	.1
0	1:1	.5
10	10:1	.9
20	100:1	.99
30	1000:1	.999

5.1.3 Links

Links determine how a space's degree of belief will be affected when the degree of belief of one or more of its antecedents changes. That is, when information entered by the user changes the degree of belief of a space, the degrees of belief of all spaces linked to that space will also be changed.

Links can take one of three forms.

5.1.3.1 **Logical links** - define the degree of belief of one space completely in terms of one or more other spaces. Such links can be specified with AND, OR, or NOT operators, depending upon how the database designer wishes the spaces to interact. For example, a knowledge-base for assigning part numbers to new computer products might contain one space indicating that the product is a piece of hardware and another space indicating that the product is a piece of software. Assuming that each piece is either a hardware or a software product, the system only needs to ask one question, "Is the new product a piece of hardware?" Not only would the degree of belief of the first (hardware) space be defined by the user's response to this question, but, via a logical NOT link, the degree of belief of the second (software) space could also be defined, because the hypotheses that the two spaces represent are, by definition, mutually exclusive.

Logical links work as follows.

- If a Space A is defined by a logical AND link to Spaces B and C, the degree of belief for Space A will be the minimum of the degrees of belief for Spaces B and C.
- If a Space A is defined by a logical OR link to Spaces B and C, the degree of belief for Space A will be the maximum of the degrees of belief for Spaces B and C.
- If a Space A is defined by a logical NOT link to Space B, the degree of belief for Space A will be -1 times the degree of belief for Space B.

5.1.3.2 **Bayesian links** - are based on the Bayesian decision construct described in Section 2.3. A Bayesian link between two spaces is defined by a weight with which the evidence for the first space also supports the second space. The difference between a logical link and a Bayesian link is that the former changes the degree of the second space to a specific value, while the latter changes the degree of the second space by an amount determined by the weight of the link and the certainty of the user's response. (Response certainties are discussed in Section 4.2.3.)

Bayesian links can be either positive or negative. If the link is positive, it indicates that positive evidence for the truth of the first space increases the probability that the second space is true (the degree of belief becomes more positive). If the link is negative, it indicates that such evidence decreases the probability that the second space is true (the degree of belief becomes more negative).

For example, suppose that Goal Space B has a Bayesian link to Question Space A defined by a positive weight (pw) of +5 and a negative weight (nw) of -8. Further suppose that the current degree of belief of Goal Space B is -12. (See Figure 1.) When the question defined by Space A is asked, the degree of belief of Space B changes as follows.

- If Question A is answered in the affirmative with a certainty of 100%, the entire value of the positive weight is added to the current degree of belief of Space B. The degree of Space B therefore goes from -12 to -7, indicating that the probability that Space B is true has increased.
- If Question A is answered in the negative with a certainty of 100%, the entire value of the negative weight is added to the current degree of Space B. The degree of Space B therefore goes from -12 to -20, indicating that the probability that Space B is true has decreased.

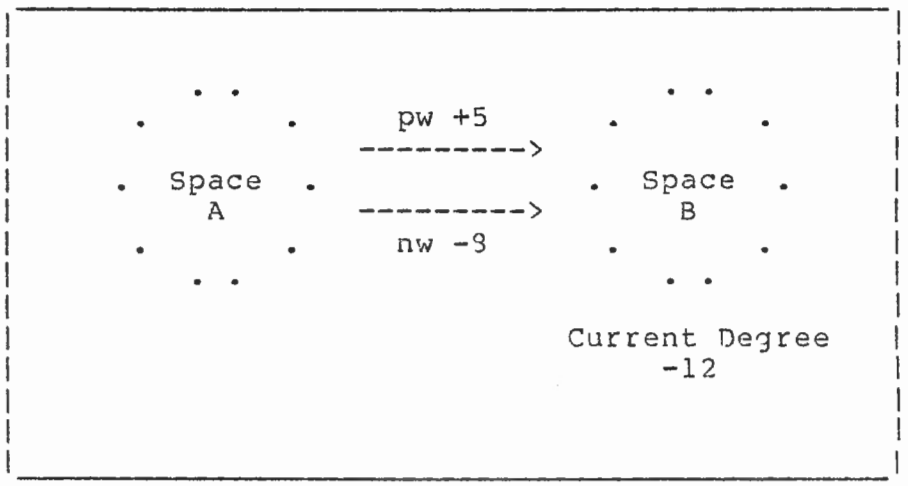

Figure 1

BAYESIAN LINK BETWEEN TWO SPACES

- If Question A is answered with a certainty of less than 100% (see Section 4.2.3), the value of the weight added to the current degree of Space B is interpolated accordingly.

5.1.3.3 **Context links** - are used to deal with situations in which it makes no sense to ask one question unless another has been answered in a certain way. For example, one AL/X knowledge-base for diagnosing car problems uses a context link between a question on revving the engine and starting the car: it is pointless to ask if it is difficult to rev the engine unless one knows that the car will start.

If Space B is linked to Space A via a context link, two conditions must exist before the hypothesis represented by Space B will be investigated:

- all evidence for Space A must have been collected, and
- the degree of belief for Space A must be in the range specified by the context link by Space B.

5.2 The AL/X Driver Program

The AL/X driver program

- reads an AL/X knowledge-base,
- constructs an internal representation of the coded knowledge-base in the form of an inference network,
- investigates network goals by asking questions of the user,
- analyzes user responses to questions and propogates this evidence through the network by changing the degrees of belief of all spaces with links to the question space, and
- provides "advice" to the user by identifying those goals whose degrees of belief have exceeded a predefined, positive probability.

5.2.1 Selection of Goal to be Investigated

AL/X always tries to collect evidence for the goal has the greatest probability of being true. After initialization, therefore, the program first investigates the goal with the highest prior degree of belief. (In most cases, this is the goal with the smallest negative degree of belief.)

It investigates this goal by asking the question that can have the greatest impact on the goal, i.e., that antecedent whose link will change the degree of belief of the goal the greatest. Once this answer is processed, AL/X applies the same criteria to select the next question. When all questions with links to the goal being investigated are asked, the program selects the next uninvestigated goal with the greatest probability of being true and begins asking questions that impact that goal.

5.2.2 Construction of Questions

AL/X questions are constructed from text associated with each space in the knowledge-base. The basic form of each question is

"How certain are you that ..."

Thus, if the text for a space is "the car is difficult to start", the question constructed from this text will be "How certain are you that the car is difficult to start?"

5.2.3 Analysis of User Responses

The user responds to questions by entering a number between -5 and +5, inclusive, indicating his or her degree of certainty that there is evidence for the hypothesis posed by the question.

- An entry of +5 indicates absolute certainty that the hypothesis is true and changes the degree of belief of the question space to the maximum value, +100.
- An entry of -5 indicates absolute certainty that the hypothesis is false and changes the degree of belief of the question space to the minimum value, -100 [3].
- An entry of 0 indicates that the user has no evidence to confirm or deny the hypothesis and leaves the degree of belief of the question space unchanged.

If the user enters a value between -5 and +5, AL/X changes the degree of belief of the question space by an interpolated amount.

However a question is answered, AL/X propogates that information throughout the network via the links defined in the knowledge-base. If this action causes the degree of belief of any goal to exceed a predefined criterion (initially 0.0 or 50% probability), that goal is flagged and its text is displayed for the user.

---

[3] To simplify matters, the system accepts entries of "y" and "n" as equivalents to +5 and -5.

5.3 A Sample AL/X Application

In an attempt to gain a thorough understanding of AL/X, Stew Kranz and I developed a sample knowledge-based AI application. The sections below

- describe the problem that we set out to solve,
- explain how we conceived the database and computed the values needed by the program,
- provide a partial listing of the coded knowledge-base,
- show part of a sample run, and
- offer a critique of the entire process.

5.3.1 Definition of the Problem

In choosing a sample problem to implement under AL/X, we decided that the problem had to be

- small enough to allow us to generate a substantial number of rules in a short amount of time [4],
- large enough to sufficiently test some of the esoteric AL/X features,
- appropriate for a knowledge-based system (that is, we needed a problem for which users would not answer all questions with 100% certainty),
- in some way relevant to Educational Services.

The problem that we developed was designed to address the following scenario.

> Each Monday morning, hundreds of students come to Bedford and face a single problem: given that they know what course they are here to take, they must determine in which classroom that course is being taught. In most cases, this problem is easily solved in one of two ways: either students find their courses listed on the classroom assignment board, or they ask the Registrar which classroom they should report to.

---

[4] We learned from others that rule production is by far the most difficult and time-consuming task in setting up a knowledge-based system. The AL/X authors claim that it took them 12 months to develop the AL/X driver program and 18 months to develop a 60-rule database.

> Before coming to Bedford, all students are issued a form that indicates the name of the course for which they are registered. Most students bring this form with them on the first day of class. Let us imagine that a student forgets to bring this form and, unfortunately, has forgotten the name of the class that he or she is here to take. Looking at the classroom assignment board provides little help, for the student is sufficiently computer-naive to be unable to distinguish between "TOPS-10 ALP", "TOPS-20 ALP", and "TOPS-20 DBMS".
>
> The student therefore proceeds to the Registrar and explains his or her problem. The Registrar tries to help the student by asking a series of questions designed to identify for which course the student is registered. (Assume that no cross-listing of course assignments by student is available.) The registrar might begin by asking if the student's course pertains to hardware or software. Depending upon the student's familiarity with computer terms, he or she will answer these questions with various degrees of certainty, using phrases such as "I think so", "No, that doesn't sound familiar", and "Yes, that's it!", until the correct course name is found and the Registrar can then assign the student to the proper classroom.

Given the large number of courses and topics covered by Educational Services' curriculum and the varying degree of overlap between these courses, construction of a meaningful knowledge-base with realistic positive and negative Bayesian links was not difficult. In addition, the phrases described in the last paragraph above lend themselves perfectly to the -5 to +5 response certainty scheme used by AL/X.

5.3.2 Designing the Knowledge-Base

The knowledge-base was designed from classroom assignment data for the week of 9 November 1981. This data took the form shown in Table 2.

Each of the 31 classrooms listed in Table 2 was defined as a goal. To gather evidence on which course the student was registered for, a number of descriptors (characteristics or keywords) were chosen for each course. Forty-four such descriptors were chosen, and these are shown in Table 3.

The final step in designing the knowledge-base was to link the goals to the questions. This was done by constructing a matrix showing which descriptors pertained to which course. This matrix is shown in Table 4. Each "x" in the matrix indicates where a descriptor pertains to a classroom. These points are reflected

Table 2

CLASSROOM ASSIGNMENT DATA USED FOR CONSTRUCTION
OF THE SAMPLE AL/X KNOWLEDGE-BASE

Number of Classrooms = 31
Total Enrollment = 351

Room No.	Enroll-ment	Course Name
B206	14	PDP-11/44 Processor Maintenance
B208	16	RSTS/E BASIC2 RMS
B210	20	Introduction to Minicomputers
B212	22	VAX/VMS Operating System Internals
B213	15	TOPS-10 Assembly Language Programming
B214	16	VAX-11/780 Processor Internals
B223	3	Assembly Language Programming in VAX-11 Macro
B225	10	PDP-11/23 for the U.S. Army
B226	9	Programming in Fortran IV
B228	16	TOPS-20 Assembly Language Programming
B230	7	RT-11 Programming
B231	5	Programming in Macro-11
B232	18	RSX-11M User
B233	12	TOPS-20 Data Base Management Systems
B235	9	PDP-11/34 Processor Maintenance
B236	16	VAX/VMS Utilities and Commands
B237	10	RSX-11M Systems Programming
B238	8	TOPS-20 Administration
B240	6	RSTS/E Advanced BASIC-PLUS Programming
B241	8	DSM-11 Advanced Programming
B242	3	Programming VMS in VAX-11 Fortran/Macro
B243	5	RSX-11M System Programming
B244	11	System Dependent Features of VAX-11 BASIC
B245	7	LSI, PDP-11/03 Hardware and Internals
B246	8	VAX-11/750 Hardware Diagnostics User
B247	12	TJU16/TU16/TE16
B248	13	PDP-11/70 Processor Maintenance
B249	16	VAX/VMS System Management
B254	12	VAX/VMS Operator
B255	12	RL01/RL02
B256	12	VAX-11/750 Hardware Diagnostics User (Applicon)

Table 3

DESCRIPTORS FOR DISTINGUISHING COURSES IN KNOWLEDGE-BASE

AL/X Space	Text of Descriptor
hardware	a hardware course
software	a software course
lsi	being taught on an LSI or PDP-11/03 system
pdp11	being taught on a PDP-11 system
pdp1123	being taught on a PDP-11/23 system
pdp1134	being taught on a PDP-11/34 system
pdp1144	being taught on a PDP-11/44 system
pdp1170	being taught on a PDP-11/70 system
rt11	about RT-11
rsts	about RSTS/E
rsx	about an RSX system
rsx11m	about RSX-11M
rsx11mplus	about RSX-11M+
vax	about a VAX system
vax11750	about the VAX-11/750
vax11780	about the VAX-11/780
tops	about a TOPS system
tops10	about TOPS-10
tops20	about TOPS-20
program	a programming course
alp	on assembly language programming
fortran	on Fortran programming
sysprog	on systems programming
macro	on Macro programming
basic	on BASIC programming
dsm	on DSM programming
rms	on RMS programming
sysadmin	about system administration
sysdepen	about system dependent features
sysmanag	about system management
sysoper	about system operation
user	about using a system
utilcomm	about utilities and commands
internals	an internals course
opsysint	about operating system internals
procint	about processor internals
diag	about hardware diagnostics
tape	about tape drives
disk	about disk drives
procmaint	about processor maintenance
itm	an introductory course on minicomputers
army	a special course for the US Army
dbms	about Data Base Management Systems
graphics	about a graphics processor

Table 4

MATRIX OF DESCRIPTORS (QUESTION SPACES) AND CLASSROOMS (GOALS)

	Goal Space																														
Question Space	0 6	0 8	1 0	1 2	1 3	1 4	2 5	2 6	2 8	3 0	3 1	3 2	3 3	3 5	3 6	3 3	3 7	3 8	4 2	4 1	4 0	4 3	4 4	4 5	4 6	4 7	4 8	4 9	5 4	5 5	5 6
hardware	x	.	.	.	.	x	x	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	x	x	x	x	.	.	x	x
software	.	x	x	x	x	.	.	x	x	x	x	x	x	.	x	x	x	x	x	x	x	x	x	.	.	.	.	x	x	.	.
lsi	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.
pdp11	x	x	x	.	.	.	x	x	.	x	x	x	.	x	.	.	x	.	.	x	x	x	.	x	.	.	x	.	.	.	.
pdp1123	.	.	.	.	.	.	x	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
pdp1134	.	x	.	.	.	.	.	x	.	.	.	x	.	x	.	.	x	.	.	.	x	x	.	.	.	.	.	.	.	.	.
pdp1144	x	x	.	.	.	.	.	x	.	.	.	x	.	.	.	.	.	.	.	.	x	x	.	.	.	.	.	.	.	.	.
pdp1170	.	x	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	x	.	.	.	.
rt11	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
rsts	.	x	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.
rsx	.	.	.	.	.	.	.	x	.	.	.	x	.	.	.	.	x	.	.	.	.	x	.	.	.	.	.	.	.	.	.
rsx11m	.	.	.	.	.	.	.	x	.	.	.	x	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.
rsx11mplus	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.
vax	.	.	.	x	.	x	.	x	.	.	.	.	.	.	x	x	.	.	x	.	.	.	x	.	x	.	.	x	x	.	x
vax11750	.	.	.	x	.	.	.	x	.	.	.	.	.	.	x	x	.	.	.	.	.	.	x	.	x	.	.	x	x	.	x
vax11780	.	.	.	x	.	x	.	x	.	.	.	.	.	.	x	x	.	.	x	.	.	.	x	.	.	.	.	x	x	.	.
tops	.	.	.	.	x	.	.	.	x	.	.	.	x	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.
tops10	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
tops20	.	.	.	.	.	.	.	.	x	.	.	.	x	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.
program	.	x	.	x	x	.	.	x	x	x	x	.	x	.	.	x	x	.	x	x	x	x	x	.	.	.	.	.	.	.	.
alp	.	.	.	.	x	.	.	.	x	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
fortran	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.
sysprog	.	.	.	x	x	.	.	.	x	.	x	.	.	.	.	x	x	.	.	.	.	x	x	.	.	.	.	.	.	.	.
macro	.	.	.	.	x	.	.	.	x	.	x	.	.	.	.	x	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.
basic	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	x	.	.	.	.	.	.	.	.
dsm	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.
rms	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
sysadmin	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	x	x	.	.
sysdepen	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.
sysmanag	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.
sysoper	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.
user	.	.	.	.	.	.	.	.	.	.	.	x	.	.	x	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	x
utilcomm	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
internals	.	.	.	x	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.
opsysint	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
procint	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.
diag	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	x
tape	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.
disk	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.
procmaint	x	.	.	.	.	x	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.
itm	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
army	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
dbms	.	.	.	.	.	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
graphics	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	x

in the knowledge-base by links between the questions spaces. (The periods in the table are meaningless; they were added simply to make the table easier to follow.)

5.3.3 Computation of Numerical Parameters

The 44 descriptors were coded as question spaces and the 31 classrooms were coded as goal spaces. To complete the knowledge-base, prior degrees of belief had to be assigned to each space and the positive and negative weights of all Bayesian links had to be established.

5.3.3.1 **Prior degrees of belief for goals** - were interpreted as the probability with which any random student walking through the door was assigned to that particular class. This probability was computed using the odds/log formula

$$\text{prior degree}(H) = 10 * \log10(\text{odds}(H))$$

$$\text{where} \quad \text{odds}(H) = \frac{\text{prob}(H)}{(1-\text{prob}(H))}$$

$$\text{and} \quad \text{prob}(H) = \frac{\text{number of students in class}(H)}{\text{total number of students}}$$

Thus, for class with 15 students using a total population of 351 students,

$$\text{prob}(H) = 15/351 = 0.0427$$
$$\text{so} \quad \text{odds}(H) = 0.0427/(1-0.0427) = 0.0446$$
$$\text{thus} \quad \text{degree}(H) = -13.5$$

The values computed using this formula are shown in Table 5.

5.3.3.2 **Prior degrees of belief for questions** - were interpreted as the probability with which any random student walking through the door would be assigned to a class with that question as a descriptor. This probability was computed using the same odds/log formula, but with prob(H) redefined as

$$\text{prob}(H) = \frac{\text{number of students taken courses with space}(H) \text{ as a characteristic}}{\text{total number of students}}$$

where total number of students = 351

Table 5

COMPUTED PRIOR DEGREES OF BELIEF FOR GOALS

Number of Classrooms = 31
Total Enrollment = 351

Room No.	Enroll-ment	Prior Degree
B206	14	-13.8
B208	16	-13.2
B210	20	-12.2
B212	22	-11.7
B213	15	-13.5
B214	16	-13.2
B223	3	-20.6
B225	10	-15.3
B226	9	-15.8
B228	16	-13.2
B230	7	-16.9
B231	5	-18.4
B232	18	-12.7
B233	12	-14.5
B235	9	-15.8
B236	16	-13.2
B237	10	-15.3
B238	8	-16.3
B240	6	-17.6
B241	8	-16.3
B242	3	-20.6
B243	5	-18.4
B244	11	-14.9
B245	7	-16.9
B246	8	-16.3
B247	12	-14.5
B248	13	-14.1
B249	16	-13.2
B254	12	-14.5
B255	12	-14.5
B256	12	-14.5

The values computed using this formula are shown in Table 6.

5.3.3.3 **Weights of Bayesian links** - were computed by reasoning that the effect of a question on a goal is inversely proportional to the number of students taking courses which have that question as a descriptor. Thus, the greater the number of students taking a course with any specific question as a descriptor, the lower the positive weight of that question.

Using this logic, the effect of question space(H) on any goal can be expressed as

$$\text{prob(H)} = \frac{\text{total number of students} - \text{number of students taking courses } \textbf{with} \text{ space(H) as a descriptor}}{\text{total number of students}}$$

The positive weight of question space(H) can then be computed as

$$\text{positive weight} = \frac{\text{prob(H)}}{\text{1-prob(H)}}$$

Conversely, negative weights were computed using the number of goals which do not have that space as a descriptor. Thus, the greater the number of goals with any specific space as a descriptor, the higher the negative weight of that space.

$$\text{prob(H)} = \frac{\text{total number of students} - \text{number of students taking courses } \textbf{without} \text{ space(H) as a descriptor}}{\text{total number of students}}$$

which is mathematically equivalent to

$$\text{prob(H)} = \frac{\text{number of students taking courses } \textbf{with} \text{ space(H) as a descriptor}}{\text{total number of students}}$$

then, as before,

$$\text{negative weight} = \frac{\text{prob(H)}}{\text{1-prob(H)}}$$

The values computed using this formula are also shown in Table 6.

Table 6

COMPUTED PRIOR DEGREES AND BAYESIAN WEIGHTS FOR QUESTION SPACES

Space Name	Enrollment	Prior Degree	Positive Weight	Negative Weight
hardware	113	-3.2	2.1	-0.5
software	238	3.2	0.5	-2.1
lsi	7	-16.9	49.1	0.0
pdp11	157	-0.9	1.2	-0.8
pdp1123	17	-12.9	19.6	-0.1
pdp1134	73	-5.8	3.8	-0.3
pdp1144	78	-5.4	3.5	-0.3
pdp1170	44	-8.4	7.0	-0.1
rt11	7	-16.9	49.1	0.0
rsts	31	-10.1	10.3	-0.1
rsx	42	-8.7	7.4	-0.1
rsx11m	37	-9.3	8.5	-0.1
rsx11mplus	14	-13.8	24.1	0.0
vax	128	-2.4	1.7	-0.6
vax11750	109	-3.5	2.2	-0.5
vax11780	108	-3.5	2.3	-0.4
tops	51	-7.7	5.9	-0.2
tops10	15	-13.5	22.4	0.0
tops20	36	-9.4	8.8	-0.1
program	148	-1.4	1.4	-0.7
alp	34	-9.7	9.3	-0.1
fortran	12	-14.5	28.3	0.0
sysprog	87	-4.8	3.0	-0.3
macro	42	-8.7	7.4	-0.1
basic	33	-9.8	9.6	-0.1
dsm	8	-16.3	42.9	0.0
rms	16	-13.2	20.9	0.0
sysadmin	36	-9.4	8.8	-0.1
sysdepen	11	-14.9	30.9	0.0
sysmanag	16	-13.2	20.9	0.0
sysoper	12	-14.5	28.3	0.0
user	54	-7.4	5.5	-0.2
utilcomm	16	-13.2	20.9	0.0
internals	45	-8.3	6.8	-0.1
opsysint	22	-11.7	15.0	-0.1
procint	23	-11.5	14.3	-0.1
diag	20	-12.2	16.5	-0.1
tape	12	-14.5	28.3	0.0
disk	12	-14.5	28.3	0.0
procmaint	52	-7.6	5.8	-0.2
itm	20	-12.2	16.5	-0.1
army	10	-15.3	34.1	0.0
dbms	12	-14.5	28.3	0.0
graphics	12	-14.5	28.3	0.0

5.3.4 Sample Run

AL/X is invoked on VAX/VMS with an indirect DCL command file, ALX.COM. Parameter P1 in the command line specifies the name of the AL/X knowledge-base source file to be used (the extension .ALX is appended if no file extension is specified). In the command line below, the user specifies that he or she wishes to work with knowledge-base WICHCLS04.ALX.

```
$ @alx wichcls04
```

AL/X identifies itself and its version number and date and then prints the name of the knowledge-base it is accessing and the knowledge-base's version number.

```
AL/X Version 2.1, July 8th, 1931

Model Which_Class  Version 4
```

AL/X selects the goal with the greatest prior degree of belief to be the first goal to be investigated.

```
The current goal is whether or not
Your class is "VAX/VMS Operating System Internals" in Room
B212.  (b212)
Current degree is -11.7.
```

Questions that bear on the selected goal are then asked of the user. The user responds to each question by entering a number between -5 (hypothesis is false with 100% certainty) and +5 (hypothesis is true with 100% certainty) as described in Section 4.2.3 of this report. (Ellipses in the question text are an abbreviation for "How certain are you that".)

```
How certain are you that your course is a software course ?
-3
How certain are you that your course is an internals
course ?
2
... that your course is about a VAX system ?
4
... that your course is about the VAX-11/730 ?
0
```

These responses have the following meanings:

```
-3 = the user is quite sure that the hypothesis is false.
 2 = the user is somewhat sure the hypothesis is true.
 4 = the user is very sure the hypothesis is true.
 0 = the user does not know whether the hypothesis is true
     or false.
```

After all relevant questions for the current goal have been asked, AL/X prints a message to that effect.

```
There are no more significant questions for the current goal
```

The system then identifies any goals whose degrees of belief have become greater than or equal to zero, signifying that the probability that each of these goals is true is now greater than or equal to 50%. At this point, no goals meet this criterion.

```
**********************************************************
Investigated goals with degree of belief >=  0.0 are:
... none at the moment

**********************************************************
```

AL/X now selects another goal to be investigated, once again selecting that goal whose degree of belief is greatest. The degree considered this time, however, is the current degree rather than the prior degree.

```
The current goal is whether or not
Your class is "VAX-11/780 Processor Internals" in Room B214.
(b214)
Prior degree was -13.2. Current degree is -2.5.
```

The system asks questions that impact this goal and the user responds as before, but using "y" as the alternate form of +5 and "n" as the alternate from of "-5".

```
How certain are you that your course is about processor
internals ?
y
... that your course is about processor maintenance ?
n
```

After all relevant questions for this new goal have been asked, the system determines that the hypothesis represented by this goal is probably true. A message to that effect is printed for the user.

```
After considering all significant questions,
The degree that
Your class is "VAX-11/780 Processor Internals" in Room B214.
(b214)
initially was -13.2.  It is now  11.5.
```

The standard message is displayed indicating that the system has finished investigation of the current goal and the system informs the user of all goals whose degrees of belief are greater than or equal to zero.

```
There are no more significant questions for the current goal
```

```
**********************************************************
Investigated goals with degree of belief >=  0.0 are:

Your class is "VAX-11/780 Processor Internals" in Room B214.
(b214)
Prior degree was -13.2. Current degree is  11.5.

Other goals with degree >=  0.0 are:

Your class is "LSI, PDP-11/03 Hardware and Internals" in
Room B245.  (b245)
Prior degree was -16.9. Current degree is  6.3.

**********************************************************
```

Note that the degree of belief of goal B245 now exceeds zero even though this goal has not been investigated directly by questioning. This result demonstrates that evidence supplied by the user (question responses) was propogated throughout the knowledge-base, effecting all spaces with linked to the questions rather than just the goal space being investigated.

Even though AL/X has determined that one or even two goals have better than chance probability of being true, it will continue to investigate additional goals until all goals have been investigated or the user terminates the program. Thus, the system selects for questioning that uninvestigated goal whose current degree is greatest.

```
The current goal is whether or not
Your class is "LSI, PDP-11/03 Hardware and Internals" in
Room B245.  (b245)
Prior degree was -16.9. Current degree is  6.3.
```

The first question is asked, but this time the user responds with "L", an option that causes the system to print a log of the current state of the knowledge-based in a file.

```
How certain are you that your course is being taught on an
LSI or PDP-11/03 system ?
L
```

This file is shown in Table 7. Note in the first section of the table that the current degrees of virtually all goal spaces have changed from the prior degrees even though only six questions have actually been asked. Once again, this result demonstrates the power of the propogating evidence throughout the inference net.

In the third section of the table, the statuses of questions are displayed. Note that many of these questions are marked "asked" and followed by a value of 0.0 even though these questions were

Table 7

"LOG FILE" SHOWING INTERMEDIATE STATE OF THE AL/X KNOWLEDGE-BASE

Space ID's	Prior Degree	Current Degree	Question Statuses
Goals			
b206	-13.8	-11.9	
b208	-13.2	-15.3	
b210	-12.2	-14.3	
b212	-11.7	-5.3	
b213	-13.5	-15.6	
b214	-13.2	11.5	
b225	-15.3	-13.2	
b226	-15.8	-16.2	
b228	-13.2	-15.3	
b230	-16.9	-19.0	
b231	-18.4	-20.5	
b232	-12.7	-14.8	
b233	-14.5	-16.6	
b235	-15.8	-13.9	
b236	-13.2	-13.6	
b223	-20.6	-21.0	
b237	-15.3	-17.4	
b238	-16.3	-18.4	
b242	-20.6	-21.0	
b241	-16.3	-18.4	
b240	-17.6	-19.7	
b243	-18.4	-20.5	
b244	-14.9	-15.3	
b245	-16.9	6.3	
b246	-16.3	-12.5	
b247	-14.5	-12.4	
b248	-14.1	-12.2	
b249	-13.2	-13.6	
b254	-14.5	-14.9	
b255	-14.5	-12.4	
b256	-14.5	-10.7	
Intermediate Hypotheses			
hardware	-3.2	4.3	
rsx11mplus	9.3	9.3	
vax11750	3.5	3.5	
tops10	9.4	9.4	

Table 7 (con't)

"LOG FILE" SHOWING INTERMEDIATE STATE OF THE AL/X KNOWLEDGE-BASE

Space ID's	Prior Degree	Current Degree	Question Statuses	
Questions				
software	3.2	-4.3	asked	-3.0
lsi	-16.9	-16.9	not asked	
procint	-11.5	100.0	asked	5.0
army	-15.3	-15.3	not asked	
graphics	-14.5	-14.5	not asked	
rt11	-16.9	-16.9	asked	0.0
rsts	-10.1	-10.1	asked	0.0
rsx	-8.7	-8.7	asked	0.0
tops	-7.7	-7.7	asked	0.0
program	-1.4	-1.4	asked	0.0
alp	-9.7	-9.7	asked	0.0
fortran	-14.5	-14.5	asked	0.0
sysprog	-4.8	-4.8	asked	0.0
macro	-8.7	-8.7	asked	0.0
basic	-9.8	-9.8	asked	0.0
dsm	-16.3	-16.3	asked	0.0
rms	-13.2	-13.2	asked	0.0
sysdepen	-14.9	-14.9	asked	0.0
user	-7.4	-7.4	asked	0.0
utilcomm	-13.2	-13.2	asked	0.0
opsysint	-11.7	-11.7	asked	0.0
dbms	-14.5	-14.5	asked	0.0
pdp11	-0.9	-0.9	not asked	
pdp1123	-12.9	-12.9	not asked	
pdp1134	-5.8	-5.8	not asked	
pdp1144	-5.4	-5.4	not asked	
pdp1170	-8.4	-8.4	not asked	
rsx11m	-9.3	-9.3	asked	0.0
vax	-2.4	8.4	asked	4.0
vax11780	-3.5	-3.5	asked	0.0
tops20	-9.4	-9.4	asked	0.0
sysadmin	-9.4	-9.4	not asked	
sysmanag	-13.2	-13.2	not asked	
sysoper	-14.5	-14.5	not asked	
internals	-8.3	-0.4	asked	2.0
diag	-12.2	-12.2	not asked	
tape	-14.5	-14.5	not asked	
disk	-14.5	-14.5	not asked	
procmaint	-7.6	-100.0	asked	-5.0
itm	-12.2	-12.2	not asked	

not actually presented to the user. The reason for this result is that these questions all pertain to software courses. The knowledge-base therefore defined a context link between space "software" and each of the spaces being discussed, meaning that it makes no sense to ask these questions if the degree of space "software" is less than its prior degree. Since space "software" was asked and the user responded with "-3.0", the current degree of space "software" was reduce to -4.3 from a prior degree of +3.2 and all questions with context links to this question were flagged as already asked.

The user terminates the AL/X program by responding with option Q (for quit) to any question.

```
How certain are you that your course is being taught on an
LSI or PDP-11/03 system ?
Q

Would you like another consultation session with this model?
Type y or n
N
$
```

5.3.5 **Example of Knowledge-Base Coding**

A listing of part of the AL/X knowledge-base used for the sample run is shown in Table 8. Each space is identified by the keyword "space" followed by the space name. The keywords "text description" on the next line introduce the text to be associated with the space. The text itself is initiated by the delimiter "/*" and terminated by the delimiter "*/".

The keyword "inference" introduces the prior degree of the space and the definition of its links. The prior degree is expressed as a number preceded by the keyword "prior". Logical links such as "not software" are specified after the keywords "logical definition". Bayesian links are specified after the keywords "rules antecedents" (see space b214). Each space that impacts the space being defined is listed, followed by a numerical value for the positive weight (pw) and second value for the negative weight (nw). Note that both forward and backward references are allowed in the database with no explicit flagging. This explains why the rules can be "shuffled" as explained in Section 3.1 of this report without affecting the functioning or efficiency of the inference network.

The keyword "control" introduces context links and other, more esoteric, knowledge-base structures. Context links are specified by listing those question spaces that are dependent upon the space being defined. The exact way that the space being defined must be answered to make each dependent space meaningful, i.e., the required range for the current space's degree of belief,

Table 8

EXAMPLE OF AL/X KNOWLEDGE-BASE CODE

```
space hardware
    text description /* your course is a hardware course */
    inference prior -3.2
              logical definition not software
    control context for lsi       int prior max
            context for procint   int prior max
            context for army      int prior max
            context for graphics  int prior max

space vax
    text description /* your course is about a VAX system */
    inference prior -2.4
    control context for vax11750  int prior max
            context for vax11780  int prior max

space vax11780
    text description /* your course is about the VAX-11/780 */
    inference prior -3.5

space internals
    text description /* your course is an internals course */
    inference prior -8.3
    control context for opsysint  int prior max
            context for procint   int prior max

space procint
    text description /* your course is about processor
                        internals */
    inference prior -11.5

space procmaint
    text description /* your course is about processor
                        maintenance */
    inference prior -7.6

space b214
    text description /* Your class is "VAX-11/780 Processor
                        Internals" in Room B214. */
    inference prior -13.2
              rules antecedents ( hardware   pw  2.1  nw  -0.5
                                  vax        pw  1.7  nw  -0.5
                                  vax11780   pw  2.3  nw  -0.4
                                  internals  pw  6.8  nw  -0.1
                                  procint    pw 14.3  nw  -0.1
                                  procmaint  pw  5.8  nw  -0.2 )
```

follows the dependent space name in coded form. For example, "int prior max" means that the degree of the current space must be within the interval of its prior degree and the maximum value (+100) for the dependent space to be meaningful.

6.0 A CONCEPTUAL LOOK AT A RULE-BASED TUTOR

The British Open University is developing what they call a "rule-based tutor", a knowledge-based AI system to match CAI strategies to models of student behavior. This work is interesting from both an AI perspective and the perspective of the type of instruction that takes place at the Open University.

6.1 Profile of The British Open University

The latter perspective is important because the Open University probably provides the best model of where industrial training is going in the next decade: towards distributed education. The Open University has virtually no on-campus classes and no resident undergraduate student population. All teaching is done via broadcast television using the broadcast facilities of the BBC. Students watch lectures and demonstrations on television, perform learning activities specified in study guides, and correspond with instructors by mail. The University offers both undergraduate and graduate programs with no residence requirement.

For some things, of course, students must gain access to University facilities. In computer science courses, for example, students must get some time on-line. This is done in several ways. Computer centers that are networked to the University's central teaching computers are located throughout the country, and students can also dial in to the University's computers. (The University has a DECsystem-20/60, two 20/40s, two 11/60s, and Terak and Acorn microcomputers.) About 30,000 students use the University's computer services each year. In addition, 3000 students have microcomputers in their homes (on loan) for use in certain hardware courses.

With this extensive student population as a background, the Institute of Educational Technology (an autonomous affiliate of the University) has formed a CAI research group consisting of 20 academic members from various University departments (6 of the 20 are from the Institute of Educational Technology). This research group has three "strands":

- an Evaluation Strand that is focusing on evaluation of student achievement and the effectiveness of teaching programs,
- a New Technology Strand this is experimenting with new methods for the delivery of instruction [5], and
- an AI Strand that is working on student models and knowledge-based systems.

The remainder of this section describes work going on as part of the AI Strand.

6.2 O'Shea's Rule-Based Tutor

The system described below does not exist. It was conceived by Tim O'Shea (1979), Director of the Institute of Educational Technology. The purpose of the system is to specify CAI branching strategies via AI production rules, and therefore create a CAI system that is highly adaptive to a number of student models.

6.2.1 Teaching Scenario

Consider a scenario in which there are a number of teaching operations. (These would be equivalent to "modules" in DEC terms.) Let these teaching operations be represented by the symbol T(i), where $1 \leq i \leq 3$.

Each T(i) teaching operation is designed to result in a student acquiring an associated concept C(i). The probability that a student acquires the associated concept is somewhat less than 1, so each T(i) teaching operation concludes with a test to determine the student's new knowledge state.

Let concept C(3) be the terminal objective of the entire teaching task (all three T(i) teaching operations). The acquisition of C(1) or C(2) is usually, but not always, a prerequisite for acquiring C(3). The relationships between teaching operations and associated concepts is shown in Figure 2.

Before beginning any of the teaching operations, each student takes a pretest and achieves a score PS, where $0 \leq PS \leq 10$. Prior experience shows that students with high PS pretest scores are more likely to acquire C(1) via T(1) than C(2) via T(2). Conversely, students with low pretest scores are more likely to acquire C(2) via T(2) than C(1) via T(1).

As a final parameter, assume that for some practical reason there is a limit of 10 teaching operations per student.

---

[5] The New Technology Strand is currently experimenting with "Cyclops", a system for delivering pictures as well as voice over standard phone lines. There are currently 100 "Cyclops boxes" in the U.K., and this group is producing materials for such distribution in their own studio.

6.2.2 State-Vector

Given this scenario, a state-vector can be constructed that models the student's current state. The model consists of four parameters and is expressed as follows.

(LASTOP, STATE, SCORE, NUMOP)

where LASTOP = **last** teaching **op**eration, T(i)
STATE = current **state** of student knowledge, C(i)
SCORE = pretest **score**, PS
NUMOP = **num**ber of teaching **op**erations

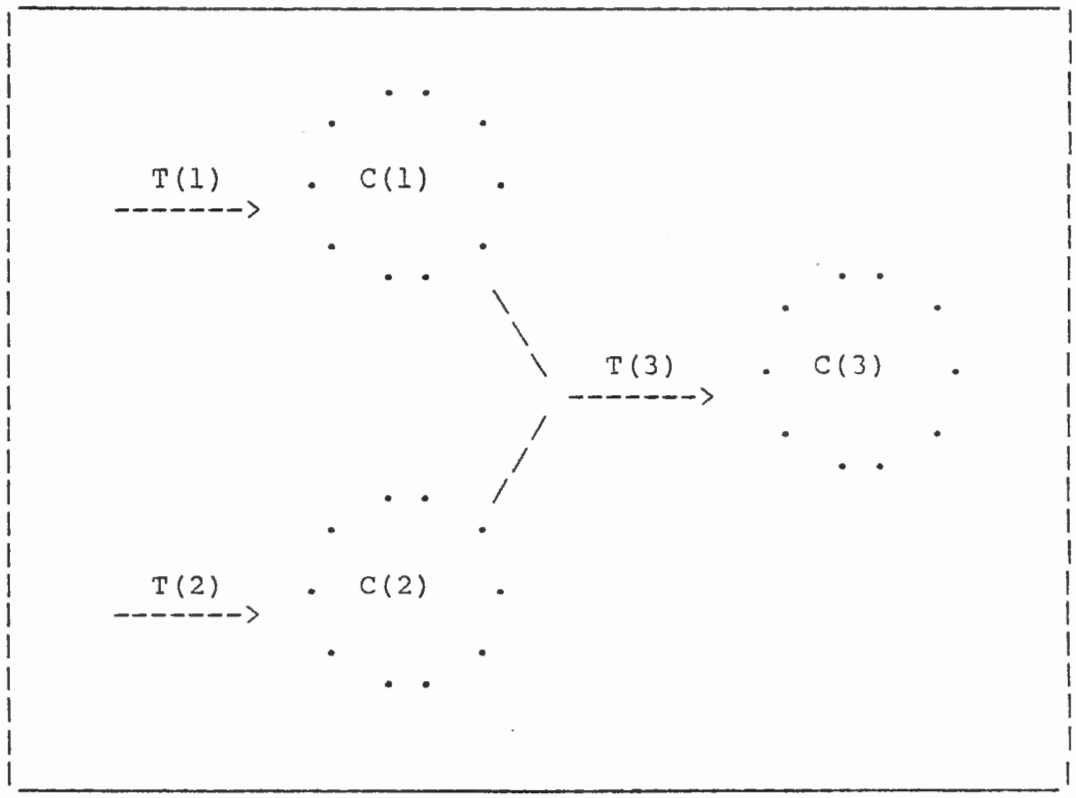

Figure 2

RELATIONSHIPS BETWEEN TEACHING OPERATIONS
AND ASSOCIATED CONCEPTS FOR O'SHEA'S RULED-BASED TUTOR

6.2.3 Partitions

The values of LASTOP, STATE, SCORE, and NUMOP are determined by the following partitions.

```
LASTOP -> LASTOP1 if LASTOP = T(1)
       -> LASTOP2 if LASTOP = T(2)

 STATE -> STATE1 if STATE = C(1) or STATE = C(2)
       -> STATE2 if STATE = C(3)

 SCORE -> SCORE1 if SCORE ≤ 5
       -> SCORE2 if SCORE > 5

 NUMOP -> NUMOP1 if NUMOP > 4
          NUMOP2 if NUMOP > 7 and NUMOP < 10
          NUMOP3 if NUMOP = 10
```

6.2.4 Ordered Condition-Action Rules

Using the values specified in the partitions, one can construct a number of condition-action rules that specify what actions the system is to take for specific combinations of the partition values. These rules are shown below and are used as follows.

The system tries to match the current state-vector to the model state-vectors on the left side of the "->" sign. Each "*" is a "wild card", so any value of the state-vector in that position generates a match. When a complete model state-vector matches the current state-vector, the action on the right side of the "->" sign takes place. These actions instruct the program to stop or to perform some teaching operation. The condition-action rules are processed in order. Therefore, when more than two rules match the current state-vector, the first match yields the action.

```
1.  (   *   , STATE2,   *   ,   *   )  ->  STOP
2.  (   *   ,   *   ,   *   , NUMOP3)  ->  STOP
3.  (   *   , STATE1,   *   ,   *   )  ->  T(3)
4.  (LASTOP1,   *   , SCORE1, NUMOP1)  ->  T(2)
5.  (LASTOP2,   *   , SCORE2, NUMOP2)  ->  T(1)
6.  (   *   ,   *   , SCORE1,   *   )  ->  T(1)
7.  (   *   ,   *   , SCORE2,   *   )  ->  T(2)
```

These condition-action rules should be interpreted as follows.

1. If the student is in STATE2, regardless of the values of the other partitions, stop the program.

2. If the student has completed 10 teaching operations, regardless of the values of the other partitions, stop the program.

3. If the student is in STATE1, regardless of the values of the other partitions, perform teaching operation T(3).

4. If the last teaching operation the student completed was T(1) and his or her pretest score was less than or equal to 5 and he or she has completed at least 4 teaching operations, perform teaching operation T(2) regardless of the student's state.

5. If the last teaching operation the student completed was T(2) and his or her pretest score was greater than 5 and he or she has completed between 7 and 10 teaching operations, perform teaching operation T(1) regardless of the student's state.

6. If the student's pretest score was less than or equal to 5, regardless of the values of the other partitions, perform teaching operation T(1).

7. If the student's pretest score was greater than than 5, regardless of the values of the other partitions, perform teaching operation T(2).

Note the importance of the order in which these rules are evaluated. If Rule 6 was evaluated before Rule 4, or Rule 7 was evaluated before Rule 5, the result of the action decision would be exactly opposite the desired one.

7.0 A PARTIAL LIST OF AI EFFORTS AT DEC

Listed below are synopses of AI projects and activities that are going on within DEC. This list is not complete, and some of the synopses may be a bit dated. In addition, these synopses represent my understandings of the projects' premises and goals. They have not been reviewed by the project staffs and may therefore contain inaccuracies. However, the list certainly shows that the Corporation is making a substantial effort to develop AI expertise and use this technology to our advantage.

7.1 XCON and XSEL

The first truly production AI system at DEC was developed in conjunction with Carnegie-Mellon University. This system, called XCON, was built to automate the configuring VAX-11/780 and 11/750 computers. The value of this system is described in the Corporation's 2nd Quarter Report for Fiscal Year 1982 as follows.

> Before the computerized configuration process was developed, a technical editor had to review a customer's order to determine that all components were compatible and to identify interface hardware necessary to build a complete system. When a customer's order is handled by XCON, the computer automatically evaluates all the desired system components, verifies the system configuration, and assigns the necessary cables and other hardware required for the individual system. The customer is assured of a total packaged system, and nothing is left to chance prior to delivery.

XCON is a very large knowledge-base. According to an article in Business Week (1982), the system "contains 1200 rules and 500 descriptions of parts, engineering constraints, and specifications. To assemble one of Digital Equipment's large computer systems, XCON will run through about 1000 separate comparisons of the data".

A companion product, called XSEL, is currently under development. This second system is designed to allow salespeople in the field enter a customer's complete system order directly at regional offices.

7.2 OPS5

XCON and XSEL are written in OPS5, a rule-production language developed at Carnegie-Mellon. Charles Rehberg in Digital's Methods and Tools Group is now "the keeper of the keys" for OPS5 within DEC. He has a version of OPS5 ready for Engineering Test on VAX/VMS, and I hope to install this version on our system in

the near future. Rehberg's installation kit includes:

- an OPS5 compiler,
- an OPS5 interpreter,
- a DCL-like command line (to make calling OPS5 similar to a native VAX/VMS command),
- a modular compilation librarian,
- an OPS5 User's Manual (from Carnegie-Mellon), and
- an OPS5 User's Guide (from Rehberg).

7.3 Hardware Troubleshooter

Digital and MIT have joined forces to develop an AI system for diagnosing intrasystem hardware failures. According to an article in Computerworld (Beeler, 1981), the purpose of this system is to "speed and simplify the task of pinpointing equipment bugs in single, stand-alone configurations. The system will be particularly well-suited to diagnosing malfunctions like broken connectors, which often result in dropped bits and 'wired-together conditions,' a term that refers to the accidental connection of two or more pins in a processor's backplane".

This project is headed from DEC's side by Neil Pundit and from MIT by Randy Davis. The system is being developed in LISP, a powerful symbol manipulation language, and is based only partially on the concepts of knowledge-based systems described in this report. A.J. Chinnaswamy, a member of Jack Walden's staff (CSSE Maintainability Engineering) in Marlboro, described the reasoning behind this project and the project's goals to me as follows.

- All of our hardware systems are developed with CAD/CAM systems. Therefore, a complete, 100% accurate technical description of each system exists in an on-line database.
- By tapping this database to generate trouble-shooting rules, we hope to create a trouble-shooter that is expert along two dimensions: (1) it will always interpret test results accurately, and (2) it will always take the shortest, most efficient, path to isolating the fault while assuring that all other possible faults are also considered.

In a telephone conversation last December, Randy Davis emphasized to me that the failures they are currently working with are of the non-intermittent and reproducible variety. That is, he explained, the very simplest of failures. He obviously did not

want to set expectations too high at this point. Chinnaswamy believes that it will be 2-3 years before the goals of this project are reached.

7.4 Image Processing

Tom Williams of Corporate Research and Development in Hudson is working with image processing (or what he calls "vision processing") using a special processor that was recently purchased for this project. Tom has been able to program this system to segment a visual image into various components and identify these components with reasonable accuracy drawing on a knowledge-base of component characteristics.

7.5 Knowledge Engineering Steering Group

John Ulrich (originally of Corporate R&D and now working on the XCON project) formed a Corporate Knowledge Engineering Steering Group in July, 1981. The goals of this group are

- to obtain new knowledge engineering techniques,
- to identify problems within Digital that may be solved by knowledge engineering,
- to demonstrate the feasibility of the proposed solutions by creating working prototypes,
- to collaborate with other Digital groups to evolve the prototypes into production quality software,
- to aid in the installation of the systems into the appropriate Digital organization, and
- to evolve a plan for organization of knowledge engineering technology within Digital.

This steering group is now headed by Rick Peebles of Corporate R&D. It holds monthly meetings at which various speakers from both inside and outside the Corporation present results of AI development efforts.

8.0 AI/CAI RESEARCH EFFORTS TO BE PURSUED

8.1 Review of Additional Literature

Now that we understand the basic concepts of knowledge-based AI systems, the next step is to review literature on additional AI applications to CAI. Readers familiar with this general area may be shocked that the work of John Seely Brown, for example, is not a major section of this report. I have just begun to look at Brown's work and that of his colleagues at Stanford, Xerox PARC, and BB&N, and I hope to be knowledgeable of these applications within the next few months.

8.2 Examination of Additional Systems

I also hope to have the opportunity to examine working AI/CAI systems beginning this summer. I am planning to attend a conference on AI/CAI in early June at which Brown will be a featured speaker, and at which several working systems are scheduled to be demonstrated. I have also planned visits to AI/CAI sites pending funding for FY83.

8.3 Specification of an AI/CAI Prototype System

By the end of FY83, I hope to be able to write a specification for integrating AI techniques into Educational Services' existing CAI authoring system. At the low end, this effort might use a knowledge-base to drive our current DRAW-oriented CAI system. Such a system might replace some of the course development effort that we now refer to as "lesson programming". This effort would, in effect, form the basis for a sophisticated CAI "author language". Its main advantage over our current system would be improved control of branching.

Further down the pike, we might consider trying to develop a "generative" CAI system in which knowledge is represented by a database and students are directed through this database in varying ways depending upon their prerequisite skills, learning styles, and professional goals. At this time, I see both of these efforts as specification projects. I do not think that we would be ready to try to build a working prototype for at least another year, but perhaps some of what we learn can be used more rapidly to achieve the two goals stated in Section 1.0: to investigate CAI courseware development tools that have the potential to improve

- the quality of CAI courseware and/or
- the efficiency of the CAI course development process.

9.0 BIBLIOGRAPHY AND RELATED READINGS

Beeler, Jeffrey, 1981. "Researchers Working on AI for Diagnosing Failures in Intrasystem Hardware", Computerworld, November 16, 1981.

Business Week, 1982. "Artificial Intelligence: The Second Computer Age Begins", Business Week, March 8, 1982.

Forgy, Charles L., and **J. McDermott, 1977.** "OPS, a Domain-Independent Production System Language," in Proceedings of the Fifth International Joint Conference on AI.

Forgy, Charles L., 1981. OPS5 User's Manual. Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.

O'Shea, Tim, 1979. "Rule-Based Computer Tutors", in Expert Systems in the Microelectronic Age, ed. Donald Michie. University of Edinburgh Press, Edinburgh, Scotland.

Paterson, Andy, 1981. AL/X User's Manual. Intelligent Terminals, Ltd., Edinburgh, Scotland (photocopied).