Basic Concepts in Knowledge-based Systems

Jesse M. Heines

Digital Equipment Corporation
Bedford, Massachusetts

Abstract Knowledge-based systems are receiving an increasing amount of attention as viable tools for solving a wide range of computer programming problems. Such systems are also useful in educational environments for introducing some of the basic concepts of artificial intelligence. This paper provides a brief overview of the role of knowledge-based systems in educational environments and presents the basic concepts underlying these systems. It then takes a detailed look at one knowledge-based system, Advice Language/X (AL/X), and uses it to illustrate the concepts presented in the earlier sections.

The Role of 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, rule-based, or production 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 applica-

Machine-Mediated Learning, Volume 1, Number 1
0732-6718/83/010065-00$02.00/0
Copyright © 1983 Crane, Russak & Company, Inc. and Mentor Systems, Inc.

tions such as mechanical and medical diagnosis. Digital Equipment Corporation currently uses a large knowledge-based system (developed in conjunction with Carnegie-Mellon University) to configure VAX systems to customer specifications [1], and is developing another knowledge-based system (in conjunction with the Massachusetts Institute of Technology) to diagnose system failures in some of its simpler computer systems [2]. The most famous knowledge-based system, however, is one known as MYCIN [3], which identifies bacterial diseases from a large number of characteristics of cultures grown from sample bacteria.

In educational circles, knowledge-based systems have been used in several highly creative ways. For example, the rules that control MYCIN have been used to generate NEOMYCIN, a computer-assisted instruction (CAI) program that teaches diagnosis of bacterial diseases [4]. CAI programs with highly adaptive teaching strategies have been created using the ability of knowledge-based systems to alter their own rules and improve their performance [5]. Current research at the Bristol Open University Institute of Educational Technology is aimed at developing generalized authoring tools for integrating knowledge-based branching into industrially produced CAI training programs [6].

It is important to note that knowledge-based systems represent but one application of artificial intelligence to computer-assisted instruction. Other work represents somewhat different approaches. For example, Burton and Brown [7] at Xerox Palo Alto Research Center, Palo Alto, CA, and Stevens [8] at Bolt, Beranek, and Newman, Cambridge, MA, use pattern recognition to monitor student progress and to diagnose areas in which students are weak during loosely structured computer-based drill sessions or simulations. These applications offer promising opportunities for weaning CAI away from the lockstep programs that predominate today, but they are currently beyond the capabilities of most educational programmers and systems.

This paper discusses basic concepts in knowledge-based systems and demonstrates these concepts through a program called Advice Language/X (AL/X), a "mini-version" of MYCIN developed by John Reiter, Steve Barth, and Andy Paterson in association with

the University of Edinburgh (Scotland) and supported by British Petroleum Development, Ltd.[1]

Concepts in Knowledge-based Systems

Rules

A logic in a computer program may direct the system to access data in a database, but the decisions about how to process those data are almost invariably made by logic hard-coded in the language of the program and stored in memory during program execution. In knowledge-based 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.

Sets of rules, when interpreted, form a hierarchy of condition/action pairs known as an associative or inference network. This network can be thought of as the executable image of a knowledge-base, ready to be accessed by a driver program. Such inference networks differ from simpler constructs (such as binary trees) in that each rule may be the parent of several rules and the child 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 running a knowledge-based system 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 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 a problem, the knowledge-based system's expertise can be enhanced simply by adding new rules. Neither the driver program nor the original structure of the knowledge-base has to be altered.

Instantiation

Once a knowledge-base is loaded and an inference network has been established, an initial state of the network is defined. The driver program then begins to collect additional data about the user's 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 system's intelligence, however, is that each 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 system. 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 has propagated the effects of the information throughout the inference network, the program either makes a 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 until the user has entered all the information at his or her disposal.

Boolean vs. Bayesian Decisions

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

If A then B.

This construct is the conceptual basis 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 nonstandard 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, take your umbrella.
(*b*) If it isn't, 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; 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 will rain today.
(*c*) +30 Your spouse says it will 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 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 will not 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 in to 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 knowledge-based 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).

The AL/X System

AL/X is a relatively simple knowledge-based system. It is written in Pascal, and thus runs on a large number of standard computer systems. 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 the basic concepts of knowledge-based systems. The program authors describe the software as follows [9]:

> 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.
> 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 driver program, form an inference network of "evidence/hypothesis" relationships. The driver program acts as both a "knowledge acceptor" during construction of the inference network and as an interactive consultant to generate and explain advice.

The AL/X Knowledge-base

The source for generating an AL/X knowledge-base is a 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.

Spaces

Spaces are the major building blocks of knowledge-bases and fall into three categories: (*a*) questions, (*b*) intermediate hypotheses, and (*c*) goals.

(*a*) *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.

(*b*) *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.

(*c*) *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. At this time, the current line of reasoning is terminated, and the system may go on to investigate other goals. Goals always have antecedents, and generally they have no consequents. Thus, goals are exit points from the network.

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 hypothesis 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 knowledge-based systems. 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 threshold levels.

In AL/X, the value of a degree of belief may vary between -100 and $+100$. A value of 0 indicates that the likelihood of a hypothesis being true is equal to the likelihood 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.

Table 1
Relationships between Degrees of Belief, Odds, and Probability.

Degree of belief	Odds	Approximate probability
−30	1:1000	0.001
−20	1:100	0.01
−10	1:10	0.1
0	1:1	0.5
10	10:1	0.9
20	100:1	0.99
30	1000:1	0.999

After Paterson [8].

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

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

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

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: (*a*) logical links, (*b*) Bayesian links, and (*c*) context links.

(*a*) *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, 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 mutually exclusive. Logical links work as follows:

(*i*) 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.

(*ii*) 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.

(*iii*) 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.

(*b*) *Bayesian links* are based on the Bayesian decision construct described earlier. A Bayesian link between two spaces is defined by a weight with which the truth of the first space supports the truth of 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 proportionally, determined by the weight of the link and the certainty of the user's response.

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 of $+5$ and a negative weight of -8. Further suppose that the current degree of belief of Goal Space B is -12. When the question defined by Space A is asked, the degree of belief of Space B changes as follows:

(*i*) 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 changes from -12 to -7, indicating that the probability that Space B is true has increased.

(*ii*) 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 changes from -12 to -20, indicating that the probability that Space B is true has decreased.

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

(*c*) *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:

(*i*) All evidence for Space A must have been collected.

(*ii*) The degree of belief for Space A must be in the range specified by the context link to/from Space B.

The AL/X Driver Program

The AL/X driver program

(*a*) reads an AL/X knowledge-base,

(*b*) constructs an internal representation of the coded knowledge-base in the form of an inference network,

(*c*) investigates network goals by asking the user questions,

(*d*) analyzes user responses and propagates this evidence through the network by changing the degrees of belief of all spaces with links to the question space, and

(*e*) provides "advice" to the user by identifying those goals whose degrees of belief have exceeded a predefined probability.

In addition to providing responses to program queries, the user has a number of options that can be used at any time during the user's interactive dialogue. For example, the user can request the driver program to print the "chain of reasoning" from a goal to a

question, tell how a goal can be established, and tell why a question is being asked.

Selection of the Goal to be Investigated

AL/X always tries to collect evidence for the goal that 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 with the greatest ability to affect the goal, i.e., that antecedent whose link will change the degree of belief of the goal the most. Once this answer is processed, AL/X applies the same criteria to select the next question. When all questions with links to the goal have been asked, the program selects the next uninvestigated goal with the greatest probability of being true and begins asking questions that affect that goal.

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?"

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.[2]

- 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 propagates the 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 probability (initially 0.0 or 50%), that goal is flagged. Once all relevant questions for that goal have been asked, its text is displayed to the user.

Table 2
Classroom-assignment data used for construction of the sample AL/X knowledge-base.

Room No.	Enrollment	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)

Number of classrooms = 31; total enrollment = 351.

A Sample AL/X Application

The sections that follow describe a sample problem developed to illustrate AL/X, explain how the database was conceived and how the values needed by the program were computed, provide a partial listing of the coded knowledge-base, and show part of a sample run.

Definition of the Problem

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

(*a*) small enough to allow us to generate a substantial number of rules in a short amount of time[3],

(*b*) large enough to sufficiently test some of the esoteric AL/X features, and

(*c*) appropriate for a knowledge-based system (that is, we needed a problem for which users would not answer all questions with 100% certainty).

The knowledge-base that we developed addressed the following scenario:

> Each Monday morning, hundreds of students come to Digital's Customer Training facilities and face a single problem: given that they know what course they are 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 to which classroom they should report.
>
> Before coming to training, all students are issued a form indicating the name of the course for which they are registered. Most students bring this form with them on the first day of class. Imagine that a student forgets to bring this form and, unfortunately, has forgotten the name of the course that he or she is to take. Looking at the classroom assignment board provides little help, because 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 the course for which 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 direct the student to the proper classroom.

Given the large number of courses and topics covered by Digital's 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.

Designing the Knowledge-base

The knowledge-base was designed from classroom assignment data for the week of November 9, 1981. These 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 the course for which the student was registered, a number of descriptors (characteristics or keywords) were chosen for each course. 44 such descriptors were chosen, and these are shown in Table 3.

The final step was to link the goals to the questions. This was done by constructing a matrix showing which descriptors pertained to which course, as shown in Table 4. Each x in the matrix indicates where a descriptor pertains to a classroom. These points are reflected in the knowledge-base by links between the question spaces. (The periods in the table are meaningless; they were added simply to make the table easier to follow).

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.

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

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 odds/log formula:

Table 4
Matrix of descriptors (question spaces) and classrooms (goals).

Question space	Goal space																														
	06	08	10	12	13	14	25	26	28	30	31	23	35	36	23	37	38	42	41	40	34	45	46	47	48	49	54	55	56
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	.	.
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

$$\text{prior degree}(H) = 10 \times \log 10\,[\text{odds}(H)]$$

where

$$\text{odds}(H) = \frac{\text{prob}\,(H)}{[1 - \times \text{prob}(H)]}$$

and

Table 5
Computed prior degrees of belief for goals.

Room no.	Enrollment	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

Number of classrooms = 31; total enrollment = 351.

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

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

$$\begin{array}{rrcll} & \text{prob}(H) & = & 15/351 & = & 0.0427 \\ \text{so} & \text{odds}(H) & = & 0.0427/(1-0.0427) & = & 0.0446. \\ \text{Thus} & \text{degree}(H) & = & -13.5 & & \end{array}$$

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

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

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

where the total number of students = 351. The values computed using this formula are shown in Table 6.

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 that 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

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

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

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

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

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

the number of goals with any specific space as a descriptor, the higher the negative weight of that space.

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

which is mathematically equivalent to

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

then, as before,

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

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

Sample Run

After initialization, AL/X identifies itself and the name of the knowledge-base it is accessing:

AL/X Version 2.1, July 8th, 1981
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). (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 a VAX-11/780 ?
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 -2.6.
```

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

> 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 propagated throughout the knowledge-base, affecting all spaces with links 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-base 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 their prior degrees even though only six questions have actually been asked. Once again, this demonstrates the power of the propagating evidence throughout the inference net.

In the third section of the table, the status of each question is displayed. Note that many of these questions are marked "asked" and followed by a value of 0.0 even though these questions were not actually presented to the user. The reason for this result is that the system marks questions as asked if the context for asking them has been ruled out. These particular questions all pertain to software courses. The knowledge-base 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

Table 7
"Log file" showing intermediate state of the AL/X knowledge-base.

Space ID's	Prior degree	Current degree	Question status
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 (cont)

Space ID's	Prior degree	Current degree	Question status	
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	
rtll	-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	

Table 7 (cont)

Space ID's	Prior degree	Current degree	Question status	
procmaint	-7.6	-100.0	asked	-5.0
itm	-12.2	-12.2	not asked	

and the user responded with "−3.0," its degree was reduced from a prior degree of +3.2 to −4.3, and all questions with context links to this question were therefore marked as 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 PDP-11/03 system ?
Q

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

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 affects the space being defined is

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  -0.6
                                  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 )
```

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" 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. Following the dependent space name is the specification for 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. 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.

Critique and Conclusions

Knowledge-based systems offer the possibility of creating complex on-line environments in which students can freely investigate interactions between events. These systems appear to be especially applicable to discovery learning, because they have been shown to be viable tools for creating sophisticated simulation exercises.

The concepts involved in building knowledge-bases are within the reach of people with moderate computer expertise and some formal exposure to databases. However, each existing knowledge-based system seems to cater to a relatively narrow class of problems. Thus a large number of knowledge-based systems have been developed, and more seem to be popping up in the literature all the time. When used as they were intended, e.g., using AL/X for diagnostic problems, these systems have been shown to work very well. Deviations from the intended use, however, seem to cause as many programming headaches as are caused by conventional programming systems.

This situation is not unusual for a field in its infancy, and the same critique can certainly be made of CAI authoring languages. The power of knowledge-based systems has generated considerable excitement and fostered a number of highly creative applications. In addition, Kurita [10] has stated that such systems will form the basis of the Japanese fifth-generation machines. With this level of activity, we can expect to see constant refinements in these systems throughout the 1980s, and perhaps see one or two systems emerging as the primary knowledge-based tools by the middle of the decade.

From an educational viewpoint, one point is clear: smaller, cheaper machines have again put CAI at a crossroads. Computer power is now available at all levels of education, but typical lock-step CAI continues to fail in generating excitement and support. Knowledge-based systems might allow us to build the information banks that visionaries such as George Leonard [11] have predicted for some years now, and might help the computer achieve its full potential as an instructional tool.

Acknowledgements

The author is indebted to Andy Paterson of the University of Edinburgh, Howard Resnikoff of Harvard University, and Randy Levine of Digital Equipment Corporation for their reviews of preliminary versions of this paper; to Stewart Kranz of Digital Equipment Corporation for his assistance in defining the sample knowledge-base; and to Allan Shapiro of the University of Edinburgh for his help in programming the sample application.

Notes

1. AL/X is based on the Prospector consultant system for mineral exploration developed at Stanford Research Institute International by Richard Duda et al. [9].
2. To simplify matters, the system accepts entries of y (yes) and n (no) as equivalents to +5 and −5, respectively.
3. 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 about 12 man-months to develop the AL/X driver program and about 4–6 man-months to develop a 120-rule database (Paterson, 1982, personal correspondence).

References

1. "Artificial Intelligence: The Second Computer Age Begins," *Business Week*. Industrial Edition Number 2729, pp. 66–75, March 8, 1982.
2. J. Beeler, "Researchers Working on AI for Diagnosing Failures in Intrasystem Hardware," *Computerworld*. Vol. XV, No. 46, pp. 15, November 16, 1981.
3. R. Davis, B. Buchanan, and E. Shortliffe, "Production Rules as a Representation for a Knowledge-Based Consultation Program," Report STAN-CS-75-519, Memo AIM-266. Computer Science Department, Stanford University. 1975.
4. W. J. Clancey, and R. Letsinger, "NEOMYCIN: Reconfiguring a Rule-Based

Expert System for Application to Teaching," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, University of British Columbia, Vancouver, B.C., Canada. 1981.

5. T. O'Shea, "Rule-Based Computer Tutors," in *Expert Systems in the Microelectronic Age*. Donald Michie (ed.). University of Edinburgh Press, Edinburgh, Scotland. 1979.
6. T. O'Shea, R. Bornat, B. du Boulay, M. Eisenstadt, and I. Page. "Tools for Creating Intelligent Computer Tutors," Institute of Educational Technology, The Open University, Milton Keynes, England. 1981.
7. R. R. Burton, and J. Seely Brown. "An Investigation of Computer Coaching for Informal Learning Activities," in *Intelligent Tutoring Systems*. D. Sleeman and J. S. Brown (eds.). Academic Press, Inc, New York. 1982.
8. A. Stevens, "Techniques for Automatically Generating Explanations," in *Proceedings of an International Conference on Research in Computer-Based Education*, University of Delaware, Newark, Delaware. 1982.
9. A. Paterson, *AL/X User's Manual*. Intelligent Terminals, Ltd., Edinburgh, Scotland. 1981.
10. S. Kurita, "What to Expect from the 5th-Generation Computer." *Computerworld*, Vol. XVI, No. 24, pp. 13–23, June 14, 1982.
11. G. B. Leonard, *Education and Ecstacy*. Dell Publishing Co., New York. 1968.