Knowledge Engineering

The Applied Side of Artificial Intelligence

EDWARD A. FEIGENBAUM
Heuristic Programming Project
Computer Science Department
Stanford University
Stanford, California 94305

INTRODUCTION: SYMBOLIC COMPUTATION AND INFERENCE

This paper will discuss the applied artificial intelligence work that is sometimes called "knowledge engineering." The work is based on computer programs that do symbolic manipulations and symbolic inference, not calculation. The programs I will discuss do essentially no numerical calculation. They discover qualitative lines of reasoning leading to solutions to problems stated symbolically.

Knowledge

Since in this paper I often use the term "knowledge," let me say what I mean by it. The knowledge of an area of expertise—of a field of practice—is generally of two types: (a) *facts* of the domain—the widely shared knowledge that is written in textbooks, and in journals of a field, which constitutes the kind of material that a professor would lecture about in a class; (b) equally as important to the practice of a field is *heuristic* knowledge—knowledge that constitutes the rules of expertise, the rules of good practice, the judgmental rules of the field, the rules of plausible reasoning. These rules collectively constitute what the mathematician George Polya has called the "art of good guessing." In contrast to the facts of the field, its rules of expertise, its rules of good guessing, are rarely written down. This knowledge is transmitted in internships, Ph.D. programs, apprenticeships. The programs I will describe require, for expert performance on problems, heuristic knowledge to be combined with the facts of the discipline.

Expert Systems

The act of obtaining, formalizing, and putting to work these kinds of rules is what we call "expertise modeling." In the modeling of expertise, we construct programs called "expert systems." The goal of an "expert system" project is to write a program that achieves a high level of performance on problems that are difficult enough to require significant human expertise for their solution. The more common strategy of artificial intelligence (AI) research is to choose a highly simplified problem—sometimes called a "toy problem"—and exploit the toy problem in depth. In contrast, the problems we choose require the expertise of an M.D., or a Ph.D., or, at least, a very highly trained specialist in a field, to solve. An expert system of this type consists of only two things: a knowledge base and an inference procedure. The knowledge base contains the facts and heuristics; the inference procedure consists of the processes that work over the knowledge base to infer solutions to problems, to do analyses, to form hypotheses, etc. In principle, the knowledge base is separable from the inference procedure.

The Scientific Issues Underlying Knowledge Engineering

What are the central scientific issues of the artificial intelligence field from which this more applied research draws its inspiration? I would like to categorize these under three headings.

First is the problem of *knowledge representation*. How shall the knowledge of the field be represented as data structures in the memory of the computer, so that they can be conveniently accessed for problem solving?

Second is the problem of *knowledge utilization*. How can this knowledge be used in problem solving? Essentially, this is the question of the inference engine. What designs for the inference engine are available?

Third, and most important, is the question of *knowledge acquisition*. How is it possible to acquire the knowledge so important for problem solving automatically or at least semiautomatically, in a way in which the computer facilitates the transfer of expertise from humans (from practitioners or from their texts or their data) to the symbolic data structures that constitute the knowledge representation in the machine? Knowledge acquisition is a long-standing problem of artificial intelligence. For a long time it was cloaked under the word "learning." Now we are able to be more precise about the problem of machine learning; and with this increased precision has come a new term, "knowledge acquisition research."

This is the most important of the central problems of artificial intelligence research. The reason is simple: to enhance the performance of AI's programs, knowledge is power. The power does not reside in the inference procedure. The power resides in the specific knowledge of the problem domain. The most powerful systems we will be building will be those systems that contain the most knowledge.

The knowledge is currently acquired in a very painstaking way that reminds one of cottage industries, in which individual computer scientists work with individual experts in disciplines painstakingly to explicate heuristics. If applied artificial intelligence is to be important in the decades to come, we must have more automatic means for replacing what is currently a very tedious, time-consuming, and expensive procedure. The problem of knowledge acquisition is the critical bottleneck problem in artificial intelligence.

A BRIEF TUTORIAL USING THE MYCIN PROGRAM

As the basis of the exposition of underlying ideas, I will use a well-known program called MYCIN.* The EMYCIN system, described later, was described by William VanMelle as his Ph.D. thesis.[11] MYCIN is a program for medical diagnosis and therapy. It produces diagnoses of infectious diseases, particularly blood infections and meningitis infections, and advises the physician on antibiotic therapies for treating those infectious diseases. MYCIN conducts a consultation with its user, a physician. This physician is to be distinguished from another kind of doctor who works with MYCIN, the expert. The expert is the person who introduces rules into the MYCIN knowledge base. The user exploits these rules in a dialogue, an interactive consultation that finally terminates in a diagnosis and therapy. The consultation is conducted in a stylized form of English; the doctor never knows about the LISP program underneath. In the consultation the doctor is asked only for patient history and laboratory test results (exogenous data the computer could not possibly infer).

A program like MYCIN is using qualitative reasoning to discover a line of reasoning, leading to a result (in this case a diagnosis). We can expect that it should be able to explain that line of reasoning to the user. In fact, I believe it is necessary that expert consultative systems do so; otherwise, the systems will not be credible to their professional users.

Knowledge in MYCIN

TABLE 1 shows a piece of knowledge in MYCIN. MYCIN contains about 500 rules, about half of them for blood infections, half for meningitis infections. Each such "production rule" consists of an "if" part and a "then" part (sometimes called a "situation part" and "action part"). The "if part" defines a set of conditions of relevancy such that if each of these clauses is true, then the conclusion follows.† The rule is shown in approximately the way the expert would enter it; and exactly the way the doctor would see the rule if it were displayed. This piece of knowledge will be evoked from the knowledge base if the conditions are true, and will be built into the line of reasoning.

*Developed originally as the Ph.D. thesis of E. H. Shortliffe, Computer Science Department, Stanford University. Further developed by the Heuristic Programming Project at Stanford.[10]

†Any logical combination of the "if side" clauses can be used.

TABLE 1. A Piece of Knowledge in MYCIN

If:	(1) the infection that requires therapy is meningitis, and (2) the type of the infection is fungal, and (3) organisms were not seen on the stain of the culture, and (4) the patient is not a compromised host, and (5) the patient has been to an area that is endemic for coccidiomycoses, and (6) the race of the patient is one of: black asian indian and (7) the cryptococcal antigen in the csf was not positive
Then:	there is suggestive evidence that cryptococcus is not one of the organisms that might be causing the infection.

Inexact Inference

In MYCIN, there is a way for the expert to state to what extent, on a scale from 0.1 to 1.0, he believes the preconditions imply the conclusion: 1.0 is definition certainty; 0.9 is "very strong evidence"; 0.6 is "suggestive evidence"; and so on. These indices are combined in MYCIN with a very simple and easily explained function, yielding an index called a "cumulative certainty factor," an index of "strength of belief" in the line of reasoning.

MYCIN Diagnosis and Therapy

TABLE 2 shows a typical MYCIN diagnosis. Then MYCIN proceeds to conduct a shorter consultation with the physician about such things as the patient's known sensitivities to various types of antibiotics and the resistance shown by certain organisms to various kinds of antibiotics. MYCIN then produces a therapy recommendation such as shown in TABLE 3.

MYCIN's Line of Reasoning

MYCIN's line of reasoning is a chain of rules that concludes the (perhaps uncertain) presence of an infecting organism from laboratory test data and patient history. The line of reasoning is discovered by backward chaining. The search starts with the various possible organisms as "goals to be achieved" and terminates with the data.

TABLE 2. An Example of a MYCIN Diagnosis

```
INFECTION-1 is MENINGITIS
  +(ITEM-1)E.COLI[from clinical evidence only]
  +(ITEM-2)PSEUDOMONAS-AERUGINOSA[from clinical evidence only]
  +(ITEM-3)KLEBSIELLA-PNEUMONIAE[from clinical evidence only]
  +(ITEM-4)DIPLOCOCCUS-PNEUMONIAE[from clinical evidence only]
```

The explanation facility can exhibit selected portions of the chain as requested by the user. It can answer a variety of queries during or after the consultation dialogue, such as "Why (are you asking me for this information)?" or "How (was some particular conclusion reached)?" One interesting form of query is shown in TABLE 4. To answer it MYCIN must be able to recreate the unacceptable lines of reasoning.

MYCIN's Inference Procedure

We can remove the knowledge base of MYCIN and substitute a set of rules from another domain. That is equivalent to saying that the knowledge base and the inference procedure are separate things in an expert system. Removing from MYCIN its infectious disease diagnosis rules yields an inference "engine" which we call EMYCIN (for essential mycin or empty mycin or engine mycin).

TABLE 3. An Example of a MYCIN Antibiotic Therapy Recommendation

My preferred therapy recommendation is as follows:
In order to cover for Items ⟨1 2 3 4⟩:
Give the following in combination:
(1) AMPICILLIN
Dose: 3.5 g (28.0 ml) q4h IV [calculated on basis of 50 mg/kg]
(2) GENTAMICIN
Dose: 119 mg (3.0 ml, 80 mg/2 ml ampule) q8h IV [calculated on basis of 1.7 mg/kg] plus consider giving 5 mg q24h Intrathecal
Comments: Monitor serum concentrations
Since high concentrations of penicillins can inactivate aminoglycosides, do not mix these two antibiotics in the same IV bottle.

BUILDING A NEW SYSTEM WITH THE EMYCIN TOOL: PUFF

Combining with EMYCIN a set of rules for pulmonary diagnosis (diagnosis of lung diseases) produced a diagnostic program called PUFF.[9] In this diagnostic situation, a patient is breathing in and out of an instrument called a spirometer, producing measurements of flow of air in expiration and inhalation, versus lung volume. Data reduction is done by a PDP-11, and data interpretation is done by PUFF.‡ The PUFF report is reviewed by an expert physician, is signed if accurate, put into the patient record, and sent to the referring physician. Currently about 80% of the PUFF outputs are signed without modification. PUFF consists of about 250 production rules of the MYCIN-like type. It produces reports like that shown in TABLE 5.

‡PUFF was developed on a PDP-10 computer at Stanford in collaboration with doctors at Pacific Medical Center in San Francisco, and now runs at the hospital on a PDP-11.[11]

TABLE 4. An Example of MYCIN's Explanation Facility

USER:	WHY DIDN'T YOU GIVE TETRACYCLINE FOR E.COLI IN REC-1
MYCIN:	TETRACYCLINE was discounted for ITEM-1 (RECOMMENDATION-1) because there is evidence that this e.coli is not sensitive to it.

Another Application of the Emycin Tool

A group designing airplane wings for the United States Air Force uses a software package called the MARC package§ for finite element analysis. The package is excellent but complex. Its use requires considerable expertise. The design group wanted a MYCIN-like expert system to consult with designers on the use of the MARC package. EMYCIN was used to create the expert system SACON (for structural analysis consultant).[1] An engineer brings to SACON his design specifications for the structure. SACON produces for him an analysis plan, which he can then translate into subroutine calls for the MARC software.

TABLE 6 shows a MYCIN-like production rule for SACON. TABLE 7 exhibits a SACON output-analysis plan.

Concluding Remarks on Mycin-Like Systems

Before leaving the topic of MYCIN-like systems, let me just remark on two trends. The first is a trend in knowledge engineering to put in

TABLE 5. An Example of the Output of PUFF

INTERPRETATION: Elevated lung volumes indicate overinflation. In addition, the rv/tlc ratio is increased, suggesting a mild degree of air trapping. Forced vital capacity is normal but the fevl/fvc ratio is reduced, suggesting airway obstruction of a mild degree. Reduced mid-expiratory flow indicates mild airway obstruction. Obstruction is indicated by curvature in the flow-volume loop of a small degree. Following bronchodilation, the expired flow shows slight improvement. This is confirmed by the lack of change in airway resistance. The low diffusing capacity indicates a loss of alveolar capillary surface, which is moderate.

CONCLUSIONS: The low diffusing capacity, in combination with obstruction and a high total lung capacity would be consistent with a diagnosis of emphysema. The patient's airway obstruction may be caused by smoking. Discontinuation of smoking should help relieve the symptoms.

PULMONARY FUNCTION DIAGNOSIS;
1. MILD OBSTRUCTIVE AIRWAYS DISEASE EMPHYSEMATOUS TYPE.

§For MARC Analysis Research Corporation.

TABLE 6. A Piece of Knowledge (Production Rule) in the SACON System

If:	(1) The material composing the substructure is one of metal, (2) The analysis error (in percent) that is tolerable is between 5 and 30 (3) The nondimensional stress of the substructure is greater than .9, and (4) The number of cycles the loading is to be applied is between 1000 and 10000,
Then:	It is definite that fatigue is one of the stress behavior phenomena in the substructure.

software packages what we know about building expert systems. EMYCIN represents one of the first of these packages. There are other packages, built for different types of inference procedures. The AGE system assists in the building of inference procedures of a type called "blackboard models," first developed at Carnegie-Mellon University in the HEARSAY-2 speech-understanding project. Another package assists with knowledge representation. Called the UNIT package, it is similar to the packages KRL and KL-ONE.

Second, let me mention a unique package that facilitates teaching of knowledge in knowledge bases built for expert systems. This package, called GUIDON,¶ is capable of teaching whatever EMYCIN can reason about. Thus, GUIDON can presently teach infectious disease diagnosis and therapy, pulmonary function disease diagnosis, and the use of the MARC structural-analysis package. GUIDON consists of a set of rules for another kind of expertise, the expertise of good teachers. If you blend the rules of good teaching with the rules of good practice in a field, then you can teach well the rules of good practice in the field. This is important

TABLE 7. A SACON Output (Analysis Plan)

The following analysis classes are relevant to the analysis of your structure: General-inelastic

The following are specific analysis recommendations you should follow when performing the structure analysis:

- Activate incremental stress-incremental strain analysis.
- Model nonlinear stress-strain relation of the material.
- Solution will be based on a mix of gradient and Newton methods.
- User programs to scan peak stress at each step and evaluate fatigue integrity should be used.
- User programs to scan stresses, smooth, and compare with allowable stresses (with appropriate safety factors) should be used.
- User programs to scan deflections, calculate relative values, and compare with code limits, should be called upon.
- Cumulative strain damage should be calculated.

¶Developed as the Ph.D. thesis of William J. Clancy at Stanford University.[3]

because the rules of good practice are almost never taught explicitly! They are usually taught informally, by apprenticeship, as I have mentioned earlier.

Hypothesis Formation and Theory Formation: DENDRAL and META-DENDRAL

One of the most widely used of the expert systems of knowledge engineering is the DENDRAL system.[2] Initially, DENDRAL analyzed mass spectral data, and inferred a complete structural hypothesis (topology only) for the molecule. DENDRAL was subsequently generalized to produce a set of structural candidates from whatever constraints happened to be available in the problem—not only the mass spectral constraints, but constraints from nuclear magnetic resonance, from other spectral data like IR or UV, or any other information that the chemist happens to know about the problem. Given a set of constraints from various kinds of available data DENDRAL will produce a set of candidate structures that are the best explanations of the data.

DENDRAL's Knowledge and Method

DENDRAL's knowledge sources are shown in Table 8. DENDRAL uses a three-stage problem-solving process. The first stage is one in which constraints on solution are inferred from spectral data. Given those constraints, plus all other constraints the chemist has noted, the program generates all structures satisfying the problem-specific and the general chemical constraints. Finally it tests the candidates to choose and rank the best. This method is called a plan-generate-and-test strategy.

DENDRAL's Applications

DENDRAL has been used in thousands of chemical structure analyses. It has users in universities and industries throughout the United States, Europe, and Australia. Some operate over the international TYMNET to Stanford; others use DENDRAL on their own machines. DENDRAL has been "exported" to the United States National Institutes

Table 8. DENDRAL's Sources of Knowledge

Graph theoretic	connectivity, symmetry
Chemical	atoms, valences, stability
Spectroscopic	mass spectrometric fragmentation rules, nuclear magnetic resonance rules
Contextual	origin of sample, chemical properties, method of isolation
Judgmental	goodness of fit between predicted and observed data

of Health. The British recoded it for a PDP-10 in Edinburgh, Scotland. It is running at Lederle Laboratories in Pearl River, New York, and at other chemical and drug companies.‖ Developed in LISP, it was rewritten in BCPL for efficiency. It has also been used to teach structure elucidation in the first-year graduate course in organic chemistry at Stanford, and also to check the correctness of published structures.

Knowledge Acquisition

The knowledge acquisition bottleneck is a critical problem. How is it that chemists arrive at their rules of mass spectrometry? They derive these rules or theories by induction from laboratory experience. The META-DENDRAL program was an attempt to model the processes of theory formation.

The "meta" level, or knowledge acquisition level, of DENDRAL accepts as input a set of known structure-spectrum pairs. We have stored thousands of these in our computer. The output of META-DENDRAL is a set of general fragmentation rules of the form used by DENDRAL, viz., some particular subgraph of chemical molecule gives rise to some particular fragmentation. (*If* this subgraph occurs, *then* this fragmentation process will occur.)

META-DENDRAL's method is also a plan-generate-and-test method. The planning process is called interpretation and summarization, interpreting each spectral data point as a fragmentation, collecting evidence for similar processes and bond environments. The generation process generates a space of plausible rules (not plausible structures a la DENDRAL, but plausible rules of mass spectrometry) constrained by the evidence and by some user-supplied context. The test phase tests the plausible rules, using all the evidence—positive and negative evidence—and generalizes or specializes the rules to improve support from the evidence, seeking a better fit between rules and evidence.

In a major knowledge acquisition experiment, META-DENDRAL inferred the rules of fragmentation for a family of complex steroidal molecules whose mass spectral theory was of interest to our chemist collaborators. A total of 33 rules (covering three subfamilies) were formed, all chemically plausible and of high quality (measured in terms of the amount of input data accounted for by each).

How good is META-DENDRAL? To what extent have we succeeded in forming by machine a piece of reasonable scientific theory, i.e., a set of fragmentation rules for mass spectrometry? We chose the classical scientific route for answering that question. We wrote out the results of the experiment described above and sent the paper to a respected scientific journal, as a scientific contribution. The contribution was

‖DENDRAL has been licensed by Stanford University to Molecular Designs Inc. for development and sale to industry.

refereed and published in the journal, the standard qualification for a piece of new knowledge entering the science.

KNOWLEDGE ACQUISITION, DISCOVERY, CONJECTURING: AM

Another attempt at modeling knowledge acquisition and discovery was the development of the AM program.** AM's task is the discovery of mathematical concepts (not necessarily new to mankind, but interestingly complex for a program to have discovered).

AM begins with a set of elementary ideas in finite set theory: the idea of a set, a multiset, set equality, etc. The program contains heuristic knowledge relevant to generating new mathematical concepts, the kinds of heuristics that an expert mathematician would have. It also has heuristics for discarding the bad ideas generated to pick out the interesting new mathematical conjectures. These are the so-called heuristics of interestingness. Thus the knowledge base contains heuristics of combination ("generate") and heuristics of interestingness ("test").

The program searches a space of possible conjectures that can be generated from the elementary ideas, chooses the most interesting, and pursues that line of reasoning. As usual, the program is capable of explaining its line of reasoning. The user can interact with the program to give familiar labels to newly generated concepts, such as "call that concept 'add' "; "call that 'prime'." The program uses the label subsequently, so that the explanation trace is understandable to the human.

With its heuristics, the program searched the space-discovering concepts like list equality (a specialization of general set equality); cardinality, therefore number; add, subtract, multiply, divide; factoring and the concept of a prime; and the fundamental theorem of arithmetic (the unique factorization of numbers into primes). AM made some conjectures in number theory that were almost really new (discovered many years ago but basically unexplored).

The program eventually began exploring a bigger space than it could cope with, for reasons that are related to my earlier discussion of power and knowledge. As AM plunged deeper into number theory, its general mathematical heuristics became less powerful at controlling search. It needed more specific heuristics about number theory. But these were not given initially because of the possible claim that could be made that the program was initially biased toward discovering number theory. The program lost power as it needed the specialized knowledge that it did not have. A new project called EURISKO is exploring how a program can discover new heuristics as it invents new kinds of things (e.g., as it discovers ideas in number theory, how can it invent heuristics about number theory?).

**Developed by Douglas B. Lenat as his Ph.D. thesis at Stanford University.[5]

Two Major Principles of Knowledge Engineering

The two major principles of knowledge engineering have already been mentioned earlier and will be summarized here.

The first is that the problem-solving power exhibited by an intelligent agent's performance is primarily the consequence of its knowledge base, and only secondarily a consequence of the inference method employed. Expert systems must be knowledge rich even if they are methods poor. This is an important result and one that has only recently become well understood in AI. For a long time AI focused its attention almost exclusively on the development of clever inference methods. But the power of its systems does not reside in the inference method; almost any inference method will do. The power resides in the knowledge.

Second, experience has shown that this knowledge is largely *heuristic* knowledge—judgmental, experiential, uncertain. This knowledge is generally "private" to an expert, not because the expert is unwilling to share publicly what he knows, but because he is often unable to. ("What the masters really know is not written in the textbooks of the masters.") This knowledge can be extracted by a careful, painstaking analysis by a second party (a knowledge engineer), operating in the context of a large number of highly specific performance problems. The expertise being modeled is multifaceted; an expert brings to bear many and varied sources of knowledge in performance.

The Promise of Knowledge Engineering

There is presently considerable interest in the scientific, engineering, and industrial use of knowledge engineering techniques. The promise, recognized but barely realized to date, is threefold.

Cost Reductions

There is a possible enormous cost savings in computation and instrumentation by using these methods. Here I would like to make the case concretely, not abstractly. In signal-processing applications, involving large amounts of data with poor signal/noise ratios, it is possible to reduce computation costs by several orders of magnitude by the use of knowledge-based reasoning rather than brute force statistical methods.

One of the expert systems whose construction I supervised[7] involved the interpretation of massive amounts of signal data with very poor signal/noise ratios. The object of the program was to produce a continuously updated "situation understanding" of the objects producing the signals, their positions in space, and their velocities. Using standard signal-processing techniques of cross correlation and autocorrelation, the computational requirements far exceeded the bounds of all computation available for the problem. In the statistical technique, no use was made of a wealth of knowledge available to interpret the signal data, for example: "textbook" information of the objects as signal-generating sources; "good guesses" available to the human controllers about the "most likely" moves of the objects over considerable periods of time; previously discerned patterns of movement; the laws of physics dictating what the objects could possibly do; what neighboring observing sites had observed; and so on. This was the true symbolic "semantics" and context of the problem. The ongoing model of the situation could be inferred almost completely from this symbolic knowledge, with only occasional reference to the massive amount of signal data for hypothesis verification and for noticing changes. The expert system we built using AI's methods of symbolic inference was able to accomplish the task using (an estimated) two orders of magnitude less computation than the statistical methods required. There is an important lesson here. It makes little sense to use enormous amounts of expensive computation to tease a little signal out of much noise, when most of the understanding can be readily inferred from the symbolic knowledge surrounding the situation.

There is an additional cost saving possible. Sensor bandwidth and sensitivity are expensive. From a symbolic model it is possible, with precision, to generate a set of signal expectations whose emergence in the data would make a difference to the verification of the ongoing model. Sensor parameters can then be "tuned" to the expected signals and signal directions; not every signal in every direction needs to be searched for.

Consider the DENDRAL program described earlier. Because the DENDRAL program knew so much about chemistry in general and mass spectrometry in particular, it could solve structure problems using low-resolution data that chemists could solve at that time only by using high-resolution instruments. Low-resolution instrumentation plus knowledge-based reasoning equaled the performance of high-resolution instruments. A low-resolution instrument costs only about $5,000, while a high-resolution instrument costs about $100,000. Therefore, $5,000 plus "smarts" equals a $100,000 instrument.

The Inevitability Argument

There is a certain inevitability to knowledge engineering and its applications. The cost of the computers will fall drastically during the coming two decades. As it does, many more of the practitioners of the world's professions will be persuaded to turn to economical automatic information processing for assistance in managing the increasing complexity of their daily tasks. They will find, in most of computer science, help only for those of their daily problems that have a mathematical or statistical core, or are of a routine data-processing nature. But such problems will be rare, except in engineering, and physical science. In medicine, biology, management—indeed in most of the world's work—the daily tasks are those requiring symbolic reasoning with detailed professional knowledge. The computers that will act as "intelligent assistants" for these professionals must be endowed with such reasoning capabilities and knowledge.

The Most Important Gain: New Knowledge

The methodology that I have been describing allows a field to "get its hands on" the real knowledge of the field. The real knowledge of the field is not in the textbooks of the field. The textbooks lack the experiential, judgmental, heuristic knowledge known to the excellent practitioners of the field. When experts argue, the bases on which they argue are largely unspoken. The methodology we use gives a way of bringing heuristic knowledge to the surface and making it concrete—so that it can be discussed, so that consensus can be achieved. If consensus is not achieved, at least the alternatives to the consensus are available for examination.

In the end it may be irrelevant that a computer program is available to be an "intelligent assistant." The gain to human knowledge by making explicit the heuristic rules of a discipline will perhaps be the most important contribution to the knowledge-based systems approach.

Problems of Knowledge Engineering

Though knowledge engineering has made great advances in the last 10 years, and is witnessing the pressure toward industrial application, it faces persistent problems.

The Lack of Adequate and Appropriate Hardware

Artificial intelligence is largely experimental, not theoretical, computer science. It builds and tests systems. Its laboratory is the computer, and it is suffering from lack of adequate laboratory facilities.

Currently applied AI is machine limited. That was not the case for the first 15 years of AI. The capabilities of computers to process symbols exceeded our ability to conceive interesting ways to process them. In the last few years the field definitely has been limited by the size and power of its computers. For example, the DENDRAL program is now solving much larger and more difficult problems than we ever conceived that it would solve. System designers are always gentle to their systems; they know the computational boundaries of what is feasible, but users do not. The users have real problems that can easily exceed the computational capabilities of the systems that we provide them. Problems in the physical sciences can command any number of large computers, while an AI project is worth only a small fraction of a DEC PDP-10. The scale must be changed.

AI researchers are now discovering how to construct specialized symbol manipulation machines. These computers have not yet been built by industry because the industry does not yet perceive a widespread market. In the past we have adapted the classical computing machines for the symbol-manipulation activities that are indeed more general activities of computing. The list processing systems, particularly LISP, in which most AI work has been done, have been pasted on top of the instruction code of conventional computers. That is a mistake. We need specialized symbol-processing devices.

The silver lining on the cloud is the emergence of machines with large memories and LISP machines. A LISP machine (Massachusetts Institute of Technology 1979), is a piece of hardware and microcode that runs a version of the LISP language.[6] This provides highly efficient symbolic processing in a personal computer environment.

Lack of Cumulation of AI Methods and Techniques

The second problem is the lack of cumulation of AI methods and techniques. The AI field tends to reward scientifically irresponsible pseudoinnovation. That is, it tends to reward individuals for reinventing and renaming concepts and methods that are well explored.

How does one cumulate knowledge in a science? One way is to publish papers and hope that other people will read them and use the ideas. A more traditional way in computer science is to cumulate ideas in software packages, e.g., the cumulation of computational methods of statistics in the large statistical packages. The creation of software packages such as EMYCIN, AGE,[8] ROSIE (at the Rand Corporation), and the various knowledge-representation packages is a hopeful sign that we will solve the problem of cumulation.

Shortage of Trained Knowledge Engineers

One of the problems of knowledge engineering is the shortage of trained knowledge engineers. There is a strong and growing demand for such specialists.[4] The universities are producing very few of them, but are themselves consuming almost the entire product.

There is significant industrial demand. The Xerox Palo Alto Research Center has hired a number of artificial intelligence researchers to investigate the office automation systems of the future—electronic offices. One company servicing the oil industry, Schlumberger Ltd., is working on applying knowledge engineering methods to handle the following problem: the number of interpretations that have to be done for signals (coming from physical instrumentation of oil wells) is growing much larger, and it is expensive and difficult to train new interpretation specialists. Schlumberger is interested in replication of expertise. They want to discover what the expertise consists of and then copy it for use at their outlying sites.

Texas Instruments has established an AI group to explore educational uses of AI, and also some aspects of computer-aided design. IBM has a group in Palo Alto, California, studying the use of AI in diagnosis of computer system failures.

Digital Equipment Corporation (DEC), which already uses an expert system to custom configure its VAX-11 line of computers, is also developing a system to diagnose malfunctioning computer systems.

More recently, we have seen the founding of companies specifically for the purpose of providing expert systems to business, industry, and government users. Among these new companies are Applied Expert Systems (Cambridge, Mass.), IntelliGenetics (Palo Alto, Calif.), Kestrel Institute (Palo Alto, Calif.), Smart Systems Technology (Alexandria, Va.), and Teknowledge, Inc. (Palo Alto, Calif.). There are a number of military applications of AI that are being done now. Hence the defense contract firms are also in the market for knowledge engineers.

Are there any silver linings to this cloud of shortage of people? I think there are. One is the recognition that the AI community must create for itself the equivalent of the aerospace industry to apply its skills and methods to real-world problems. Each new application cannot be done, in the future, by the few skilled technologists at the university laboratories. AI must have an industry that is capable of performing the process and producing usable devices.

The Problem of Knowledge Acquisition

Another problem of applied AI is a critical scientific problem—the problem of knowledge acquisition. Since the power of expert systems is in their knowledge bases, successful applied AI requires that knowledge move from the heads of experts into programs. This is now a largely manual process of working together with experts. If we continue in this way we could be well into the twenty-first century before we get generally powerful intelligent agents. The process is just too slow. Therefore we seek more automatic methods for transferring and transforming knowledge into its computer representation.

We now have knowledge acquisition systems that are interactive, involving semiautomatic ways of steering the expert to the right piece of knowledge to introduce into the system. We have also done experiments in automatic knowledge acquisition, extracting knowledge directly from "nature," i.e., from data, from evidence (e.g., the META-DENDRAL program described earlier).

Thus, there are silver linings with regard to knowledge acquisition, but the problem is an extremely difficult and important bottleneck problem in this field.

The Development Gap

Finally, there is the so-called development gap. There is a lack of an orderly bureaucratic process in the research funding agencies for handling programs after they have achieved their first success as a research project.

Promising knowledge engineering projects, on whose success in application the future credibility of the field depends, have fallen, or will certainly fall, into the so-called development gap. Industries, also, should be educated so that they perceive a commercial self-interest in filling the gap.

Acknowledgments

It is fitting, in conclusion, and perhaps instructive to the reader, to acknowledge the major sources of research funds for the work of my groups, the Heuristic Programming Project and SUMEX Project at Stanford University. The primary research funding has come from two agencies over the years: the Defense Advanced Research Projects Agency (ARPA); and the Biotechnology Resources Program of the U.S. National Institutes of Health (NIH). Project support for particular programs has been granted by the U.S. National Science Foundation (NSF). Relatively recently we have received support from the U.S. National Library of Medicine, from the Office of Naval Research (ONR), from the National Aeronautics and Space Administration (NASA), from Schlumberger-Doll Research, and from the IBM Corporation.

The work described in this article has been programmed on the facilities of the SUMEX-AIM Computer Resource at Stanford. SUMEX-AIM is a national computer resource for research on the application of artificial intelligence to biology and medicine. The national users (the AIM community) access the computer over the national computer networks TYMNET and ARPANET. The facility consists of Digital Equipment Corporation central host computers (a 2060 with 9 megabytes of main memory, a 2020 with 2.2 megabytes of memory, a VAX 11/780 with 4 megabytes of memory, and 2 VAX 11/750's with 2 megabytes of memory each) and a number of personal LISP work stations (5 Xerox 1100's and 1 Symbolics LM-2). This heterogeneous computing environment is linked together and to other facilities on the Stanford campus by a 3 megabytes/second Ethernet. The research language used is LISP (including the Interlisp, MacLisp, FranzLisp, and ZetaLisp dialects).

References

1. Bennett, J. S. & R. S. Engelmore. 1979. SACON: a knowledge-based consultant for structural analysis. *In* Proceedings of the Sixth International Joint Conference on Artificial Intelligence: 47–49. IJCAI. Tokyo, Japan. (Also Wm. Kaufmann. Los Altos, Calif.)

2. BUCHANAN, B. G. & E. A. FEIGENBAUM. 1978. DENDRAL and META-DENDRAL. Their applications dimensions. Artif. Intell. **11:** 5–24.
3. CLANCEY, W. J. 1979. Transfer of rule-based expertise through a tutorial dialog. Ph.D. Thesis. Computer Science Department. Stanford, Calif. (Also Stanford University Computer Science Report 79-769.)
4. FEIGENBAUM, E. A. & P. MCCORDUCK. 1983. The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World. Addison-Wesley. Reading, Mass.
5. LENAT, D. 1976. AM: an artificial intelligence approach to discovery in mathematics as heuristics search. Ph.D. Thesis. Computer Science Department. Stanford University. Stanford, Calif.
6. Artificial Intelligence Laboratory. 1979. LISP Machine Manual. Massachusetts Institute of Technology. Cambridge, Mass.
7. NII, H. P. & E. A. FEIGENBAUM. 1978. Rule based understanding of signals. *In* Pattern-Directed Inference Systems. Waterman & Hayes-Roth, Eds.: 483–502. Academic Press, Inc. New York, N.Y.
8. NII, H. P. & N. AIELLO. 1979. AGE: a knowledge-based program for building knowledge-based programs. *In* Proceedings of the Sixth International Joint Conference on Artificial Intelligence: 645–655. IJCAI. Tokyo, Japan. (Also Wm. Kaufmann. Los Altos, Calif.)
9. OSBORN, J., L. FAGAN, R. FALLAT, D. MCCLUNG & R. MITCHELL. 1979. Managing the data from respiratory measurements. Med. Instrum. **13**(6): 330–336.
10. SHORTLIFFE, E. 1976. Computer-based Medical Consultations: MYCIN. American Elsevier. New York, N.Y.
11. VANMELLE, W. A. 1979. A domain-independent production rule system for consultation programs. *In* Proceedings of the Sixth International Joint Conference on Artificial Intelligence: 923–925. IJCAI. Tokyo, Japan. (Also Wm. Kaufmann. Los Altos, Calif.)