ARTIFICIAL INTELLIGENCE AND KNOWLEDGE BASED SYSTEMS:

ORIGINS, METHODS AND OPPORTUNITIES FOR NDE

Robert S. Engelmore

Knowledge Systems Laboratory
Computer Science Department
Stanford University

INTRODUCTION

The title of my paper covers a lot of territory, so my remarks will tend to be broad rather than deep. Some of the topics I will cover are:

- What is AI?
- A brief history of AI.
- What is Knowledge Engineering?
- What are Expert Systems?
- Why should you care?
- Examples of Expert Systems.
- Opportunities for NDE.
- Selecting appropriate problems.
- Criticisms of Expert Systems.
- Looking ahead.

WHAT IS ARTIFICIAL INTELLIGENCE?

Asking for a definition of Artificial Intelligence (AI) is like the old story of the blind men describing an elephant (Fig. 1). You can get many different definitions, depending on one's point of view. Some people would describe AI as general symbolic, i.e., non-numeric, computation. The field of expert systems, which receives a lot of attention in the media and is frequently equated there (erroneously) with AI, is certainly part of the story. Advanced robotics and Vision are major areas of research within the realm of AI. Programs have been developed that "understand" what they see, in the sense of generating a high-level description of a video image or an aerial photograph. Natural Language understanding is another major area of AI research. Some people regard the activity of understanding actual written or spoken language as the fundamental problem of AI. There's also the definition that says AI is the leading edge of computer science. In other words, as complex problems are solved successfully, they are no longer regarded as AI; they're considered instead to be part of some other field, like chemistry or mechanical engineering. Therefore, the domain of AI could be defined as the residue of unsolved problems in computer science.

Let me try to make some sense of AI by describing it in terms of different types of computing. In Fig. 2 we have a two-by-two matrix

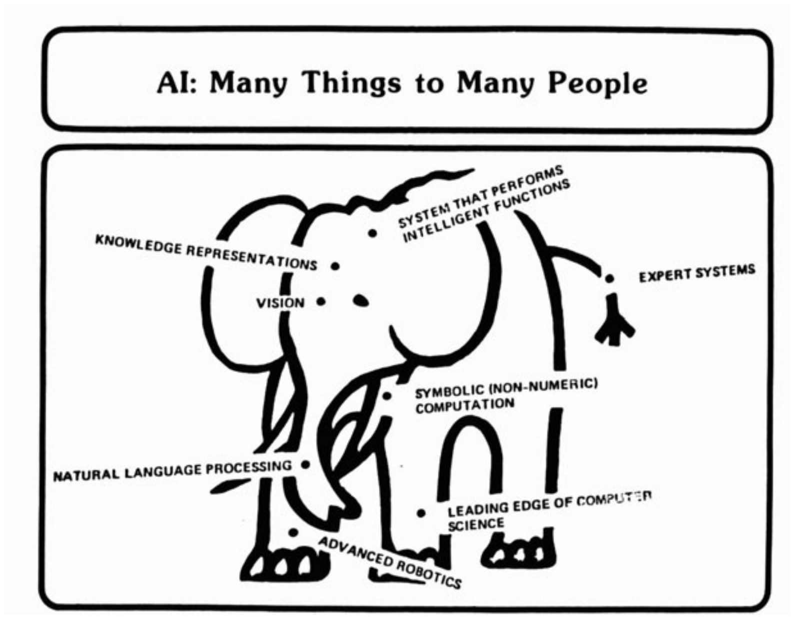
AI: Many Things to Many People
SYSTEM THAT PERFORMS INTELLIGENT FUNCTIONS
KNOWLEDGE REPRESENTATIONS
VISION
EXPERT SYSTEMS
SYMBOLIC (NON-NUMERIC) COMPUTATION
NATURAL LANGUAGE PROCESSING
LEADING EDGE OF COMPUTER SCIENCE
ADVANCED ROBOTICS

Figure 1. Views of Artificial Intelligence.

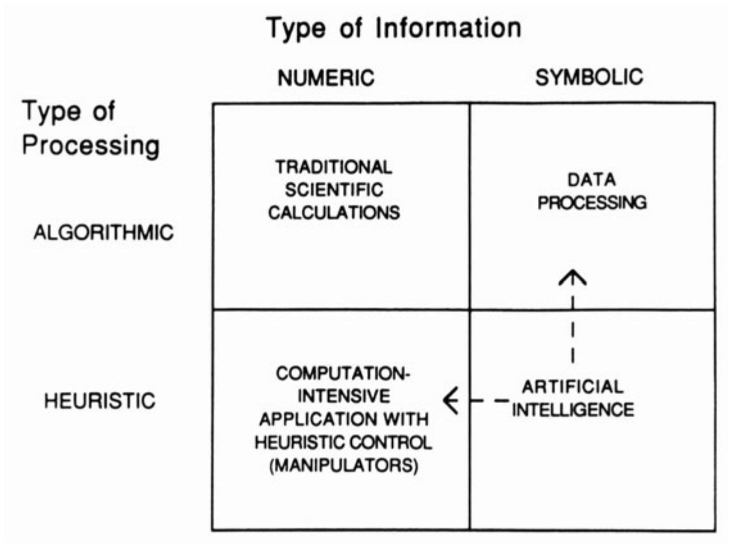

Figure 2. Types of Computing.

in which the type of processing is listed in the rows and the type of information, numeric or symbolic, with which the processing deals, is in columns. Numeric algorithmic[1] processing deals with the traditional scientific or engineering calculations with which we are all familiar. Algorithmic processing also applies to symbolic data. Retrieving information from bibliographic data bases, where the data are not numbers but text streams, is an example of this kind of activity. Here, too, the techniques for retrieving that information are very clearly defined in an algorithm, and the process is always done the same way.

[1]I use the term "algorithmic" in the sense of a fixed computational procedure that will always solve a well-defined problem having a well-defined solution.

On the bottom row is a type of processing called heuristic. Heuristics are techniques which you might think of as rules of thumb, used to solve problems by using judgmental or experiential knowledge rather than mathematical formulae. A corollary is that one is not guaranteed to find the best solution, or even any solution, but using heuristics usually works, and can be a very efficient way to reach a solution that is good enough. The term was originally coined by the mathematician Georg Polya, who defined heuristics as the art of good guessing. An example of heuristic computing with numeric informatin would be a program that must explore a large space of possible solutions, where each candidate involves extensive numeric computation to generate and/or test that solution. (Imagine all the conformations of a complex organic molecule, searching for the one conformation that best explains a set of x-ray crystallographic data.) Heuristics could be used here to guide the search, avoiding unlikely classes of solutions or enforcing the presence of certain partial solutions.

Artificial Intelligence finds its niche in the bottom right corner of the figure: heuristic symbolic computing, where the data are largely symbolic and the problems are "ill-structured", requiring heuristic techniques to reach a solution. AI makes contributions to the neighboring quadrants of the matrix. In the area of numeric heuristic processing, AI has provided techniques for searching large solution spaces efficiently. With respect to symbolic algorithmic processing, AI has linked natural language query techniques with data base retrieval, so that one can request information, using the sort of imprecise, ambiguous language that we tend to use in ordinary conversation, and let the "intelligent front end" infer what the user really asked for.

Although AI itself is primarily a field of academic research, it has spawned several subfields that are concerned with practical applications (see Fig. 3). The subfield of knowledge-based systems, which deals with building systems containing large amounts of knowledge for specific tasks, will be discussed further below. Natural language understanding is just what the name implies. In the subfield of vision and robotics, the emphasis is on building sensors and effectors for performing intelligent action. The fourth subfield is concerned with providing the hardware and software environments for building systems that manipulate and reason about symbolic information.

There are now a number of commercial companies that specialize in one or more of these subfields. Teknowledge, Inc. (Palo Alto), Intellicorp (Mt. View), Carnegie Group (Pittsburgh) and Inference (Los Angeles) are the four major AI companies that are developing knowledge-based systems and generic tools for their customers. Artificial Intelligence Corporation (Waltham) develops natural language understanding systems, and one of its products is marketed by IBM. Xerox, Symbolics, LMI and Texas Instruments are marketing computer systems specifically for AI applications.

A BRIEF HISTORY OF AI

Figure 4 (for which I thank Teknowledge for an earlier version) shows some of the main events in the history of AI. The term "Artificial Intelligence" was coined almost exactly 30 years ago by John McCarthy who has continued to be one of the field's chief theoreticians. During the first decade of its history, which we call the conception phase, the emphasis in AI research was on generality, i.e., finding general problem-solving techniques for solving symbolic problems. Two areas that provided the desired complexity for exploring and testing these general techniques were chess and symbolic logic. An early program, called LT (Logic Theorist), could prove theorems from the Principia Mathematica by Russell and Whitehead. One of the first AI programming languages, called IPL, was developed and used in the late 1950's, and GPS (General Problem Solving) was developed at Carnegie-Mellon in the early 1960s. McCarthy developed the Lisp programming language during this period, also.

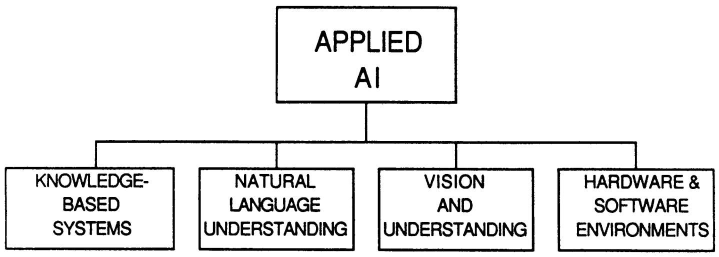
APPLIED AI
KNOWLEDGE-BASED SYSTEMS
NATURAL LANGUAGE UNDERSTANDING
VISION AND UNDERSTANDING
HARDWARE & SOFTWARE ENVIRONMENTS

Figure 3. Subfields of Artificial Intelligence.

From about 1966 to about 1973, there was a redirection away from the idea of generality. It was clear by then that one could get just so far with general (i.e., weak) problem solving methods. Most of the successes that researchers could point to were what we would today call toy problems. On the other hand, there were two projects in the mid-to-late 60s that were attacking very significant scientific problems.

One of them was the Heuristic Dendral project at Stanford, a collaborative effort of computer science, chemistry and genetics. The problem is the interpretation of mass spectrometric data, reasoning from data (a table of the masses of the molecular fragments and their relative abundance) and knowledge of analytic chemistry and mass spectrometry to derive the chemical structure of the molecule under investigation. Without going into details, let me just say that the project, which was active for over 15 years, was very successful. One measure of its success was the number of technical papers -- on the order of fifty -- that were published in refereed chemistry journals (not AI journals). The program achieved a level of expertise of a fairly advanced graduate student in chemistry. A commercial product came out of part of this work -- a program called CONGEN for generating all molecular structures from the stoichiometric formula plus various structural constraints. Dendral was one of the very first of what we now call Expert Systems.

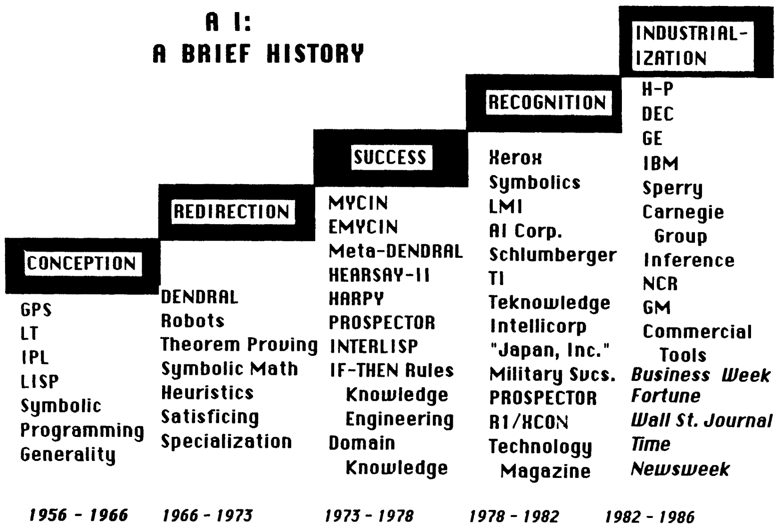

Figure 4. A Brief History of AI.

Another landmark program during this period was a program, developed at MIT, for solving mathematics problems symbolically, e.g., solving indefinite integrals. That work led to a system called MACSYMA, which is now used routinely by a large number of scientists and engineers.

The major theme during the era of redirection was the emphasis on achieving high levels of performance by using lots of specialized knowledge rather than to rely on general purpose techniques. During the next five years, this idea caught on, and there were a number of successful knowledge-based systems developed. MYCIN is by now perhaps the best known of the classic Expert Systems, and I will discuss it in more detail later. An important spinoff of the MYCIN effort was EMYCIN, the first tool for building rule-based expert systems. Meta-Dendral was a program that learned from examples the kind of knowledge that was needed for the Dendral program. It was clear early on that a major bottleneck in developing Expert Systems is the acquisition of all the knowledge that the program needs to exhibit expert performance. Meta-Dendral was one of the first systems capable of learning new knowledge automatically.

Another major research area during this period of "Success" was the Speech Understanding Program, initiated by DARPA. Two systems developed at CMU -- Hearsay II and Harpy -- achieved at least moderate performance in understanding normal speech, in a quiet room, in near real time. But perhaps more importantly, these projects produced some very interesting problem solving frameworks which have since been used in other applications. The whole field of knowledge engineering was spawned in this five-year period, when we began to understand how to develop these knowledge-based systems on something other than an ad hoc basis.

For the next four years, starting about 1978, a number of different organizations recognized the potential value of this technology, and launched their own internally sponsored AI projects. Xerox's Palo Alto Research Center had already been active in the field, and built one of the first Lisp machines that became a commercial product. Schlumberger was one of the first non-computer companies to see the value of expert systems in data interpretation, and set up their own research center. Carnegie-Mellon and Digital Equipment teamed up to produce the first expert system that went into commercial use and demonstrated real value. That was R1, a system for configuring the components of VAX computers. VAX systems are highly modular and can be ordered pretty much a la carte. Not all components are compatible, however, and there are many implicit interdependencies among components. Thus configuring one of these systems is a non-trivial task. R1, renamed XCON when it was transferred to DEC, was used on over 1000 orders per week by the end of 1983. Stanford University spun off two companies, Intelligenetics (now called Intellicorp) and Teknowledge, that specialized in knowledge engineering. The public at large became aware of this technology in the early 80s, with lots of publicity appearing in such places as Business Week, Wall Street Journal, Time Magazine.

In 1981, the Defense Science Board recognized AI as one of the most promising investments for the military. AI soon became a buzzword throughout the military services, and it seemed like all new proposal requests contained a requirement that any software must contain some artificial intelligence! Two years later DARPA,. which had been nurturing the field since the early 1970s, stepped up its support significantly when it initiated the $100M/yr Strategic Computing Program with AI as one of the core technologies.

During the past four years we have witnessed a period of increasing industrialization, and hence an increased emphasis on the applied side of AI. More new AI companies, like Carnegie Group and Inference Corp. have sprung up and are doing well, and many of the familiar names in computing, like IBM, Sperry, Digital Equipment, and Texas Instruments, now have a large investment of people and money in AI research and development. We're still a little too close to current events to identify all the milestones of the current period, but certainly one must include the arrival of low-cost workstations and PC-based knowledge systems, making the technology affordable to a large segment of the industrial and commercial sectors.

To summarize, the first decade emphasized search, i.e., the idea that you can solve any problem by search in a space (usually very large) of possible solutions and use a variety of general problem solving techniques to search efficiently. The second decade emphasized the use of application-specific knowledge, and demonstrated the power of this idea in a number of relatively small systems. The third decade, I believe, has been mainly a time of consolidation and a combining of a number of different types of problem-solving methods in order to deal with the complexity of real-world problems.

What lessons have we learned? Certainly one important lesson is that knowledge powers the solution to complex problems (it's better to be knowledgeable than smart is another way to put it). Another is that knowledge can be expressed in relatively small chunks, and that a knowledge-based system can consist of 100 or 1000 or 100,000 such chunks. And we've also learned that the key tasks in building these systems are mining the necessary knowledge and molding it so it can be used in an actual running system.

WHAT IS KNOWLEDGE ENGINEERING?

These tasks have given rise to the field called knowledge engineering, which is concerned with the acquisition, representation and use of symbolic knowledge to solve problems that normally require human attention.[2] The products of knowledge engineering are called by many names: knowledge-based expert systems, knowledge-based systems, expert systems, knowledge systems. For the purposes of this talk they are all synonymous, and I'll use the most common term, expert systems.

WHAT ARE EXPERT SYSTEMS?

We can describe expert systems on several dimensions to help distinguish them from other kinds of computer programs. One dimension has to do with methodology. Expert systems are members of the family of AI programs, and as I mentioned earlier, they emphasize symbolic information and heuristic processing. A second dimension is quality: expert systems are expected to exhibit expert-level performance in their respective application domains. A third dimension is the design dimension: Expert systems should be easy to modify or augment, and their behavior should be understandable. Finally, on the implementation dimension, a key element of expert systems is that the knowledge base is kept separate from the inference procedures, as shown in Fig. 5. In thinking about designing an expert system, one should always keep in mind the two parts: a knowledge base, which you may think of as a global data base containing facts, rules, relations, etc., and an inference engine which interprets the knowledge and controls the problem solving procedure according to some strategy. This design is in sharp contrast with most computer programs, in which the knowledge of the domain is embedded directly into the program.

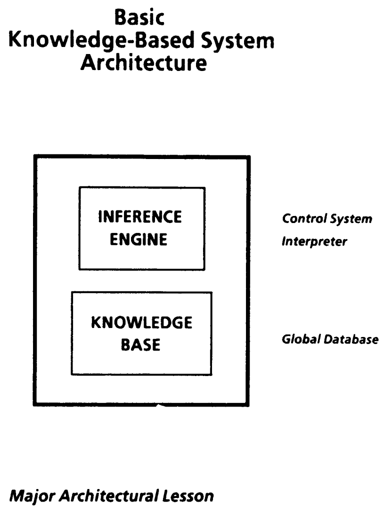

Figure 5. Separation of Knowledge and Inference Mechanism.

[2]The question naturally arises, what sort of problems that normally require human attention could or should a machine solve? Many of us like to think that all tasks can be automated up to but not including our own. As time goes on that threshold keeps moving up.

WHY SHOULD YOU CARE?

Two practical reasons for building expert systems is that they can reduce costs and/or increase the quality of many decision-making tasks. Expertise is always a scarce resource, and costly problems can arise when novices are left to solve problems on their own. Another well-known dictum is that crises and experts are almost never in the same place at the same time. Moreover, experts are not permanent: they change jobs or retire, eventually, and quality of performance drops. Expert systems can provide the expertise that is typically unavailable to less experienced personnel when or where it's needed. Moreover, experts should not have to spend valuable time on routine decision-making or bookkeeping tasks, and expert systems can be used to automate such tasks. During times of crisis, record keeping is often inaccurate or incomplete. Expert systems are able to keep consistently accurate and complete records of their actions.

EXAMPLES OF EXPERT SYSTEMS

MYCIN is a rule-based system for medical diagnosis and therapy . It was developed at Stanford about twelve years ago. It started as a Ph.D. thesis by Ted Shortliffe, who also has an M.D. degree and is now on the faculty at Stanford in the Department of Medicine. MYCIN is one of the landmark expert systems, not only because it solves an interesting problem, but also because it does it in a very clean way. By separating the knowledge from the program that uses the knowledge, MYCIN is easy to understand, replicate, and augment, and its knowledge base can be used for purposes other than diagnosis, e.g., teaching.

Why did he choose to develop this particular expert system? Shortliffe, of course, was interested in medicine. He found some doctors at Stanford who are world class experts in the diagnosis and treatment of various kinds of infectious diseases, and he focused his attention on the problem of diagnosing infecting organisms in blood and meningitis infections.

The idea behind MYCIN is that you would like to be able to make a diagnosis and prescribe an effective antibiotic treatment, using test results and information about the patient as supplied by the doctor. You would also like to be able to do it early in the treatment when you don't necessarily have all the laboratory data available. Moreover, non-expert physicians tend to overuse antibiotics, or use them irrationally, and MYCIN is intended to counteract such practice. Finally, MYCIN was intended to be an active medium for distributing expertise in the use of antibiotics, as well as the diagnosis of infectious diseases, to a broad community of practitioners who normally don't have access to experts as you might find at a university hospital.

Figures 6 and 7 show how MYCIN's diagnosis and therapy recommendations might look to a user. The diagnosis stage usually consists of identifying certain kinds of infections, and for each of these infections, identifying particular organisms that are responsible for them. It tends to be exhaustive, not just finding the first thing that might explain the data, but to find all of the possible explanations of the patient's symptoms.

The next phase, therapy, recommends a set of drugs that will cover for each of the five items in the diagnosis. In this case, MYCIN recommends some two antibiotics, prescribes the dosage, and includes comments for modification of dosage in case of certain complications such as renal failure.

How MYCIN looks to the user: Diagnosis

```
Infection-1 is Cystitis
    <Item 1> Pseudomonas-Cepacia [Organism-5]
    <Item 2> Citrobacter-Diversus [Organism-4]
    <Item 3> E-coli [Organism-3]
Infection-2 is Upper-Respiratory-Infection
    <Item 4> Listeria [Organism-2]
Infection-3 is Bacteremia
    <Item 5> Enterococcus [Organism-1]
```

Figure 6. Example of a MYCIN Diagnosis.

How MYCIN looks to the user: Therapy recommendation

```
[REC-1] My preferred therapy recommendation is as follows:
In order to cover for items <1 2 3 4 5>:
Give the following in combination:
    1: Kanamycin
       Dose: 750 mg (7.5 mg/kg) q12h IM (or IV) for 28 days
Comments: Modify dose in renal failure
    2: Penicillin
       Dose: 2,500,000 units (25000 units/kg) q4h IV for 28 days
```

Figure 7. Example of a MYCIN Therapy Recommendation.

The Knowledge base in MYCIN consists of about 400 diagnostic rules and about 50 therapy rules. A typical diagnosis rule is shown in Fig. 8. One thing to notice is that it's symbolic. There are no numbers (with one exception) in it. It has the typical "if-then" format. The conditions have to do with the site of the culture and so forth. So, this is symbolic data. Moreover, the conclusion is also symbolic. It has to do with the identity of an organism that would produce such data. The one number appearing in the rule is what is generally called a certainty factor. It's similar to a probability measure, although not used exactly that way. MYCIN has a calculus for dealing with these certainty factors so that when you chain together several rules, each of which may embody uncertain knowledge, in a line of reasoning, one can calculate an overall certainty of the conclusion.

MYCIN DIAGNOSIS RULES

IF The site of the culture is blood
The gram stain of the organism is gramneg
The morphology of the organism is rod, and
The patient is a compromised host,

THEN There is suggestive evidence (0.6) that the identity of the organism is Pseudomonas-Aeruginosa.

Figure 8. Example of a Diagnosis Rule in MYCIN.

How does MYCIN actually use these? The rule in Fig. 8 is a typical chunk of knowledge. You can see that the conclusion of this rule is a conclusion about the identity of an organism, and it's clear that this is one of the things one needs to know in order to be able to prescribe any drugs. So, one of the sub-goals of the program is to identify all of the organisms. In order for this rule to fire, each of the clauses in the IF portion must be true. The truth of the first clause, "if the site of the culture is blood," may be determined by simply asking the user, or by inferring that fact from other rules whose conclusion part says that the site of the culture is blood. To evaluate these rules, their IF portion must be evaluated, so the procedure is recursive. We call this procedure "backward chaining" because we start from a goal and work backwards by first finding the rules that conclude the goal, then generating sub-goals from the clauses on the condition side of the rule, and so forth.

Figure 9 shows a therapy selection rule. One can include justifications to the rules. These are not used by the program, but they help establish confidence in the program by indicating where these rules come from.

MYCIN: Therapy Selection Rules

IF You are considering giving chloramphenicol, and
The patient is less than 1 week old

THEN It is definite (1.0) that chloramphenicol is contraindicated for this patient.
[Justification: Newborn infants may develop vasomuscular collapse due to an immaturity of the liver and kidney functions resulting in decreased metabolism of chloramphenicol.]

Figure 9. Example of a Therapy Selection Rule in MYCIN.

One of the most powerful aspects of MYCIN is its ability to explain its line of reasoning, particularly for physicians who are understandably suspicious of using computers for such high level decision making. It is absolutely essential that the system provide not only an answer but also the ability to explain why it reached the conclusion it did. So MYCIN was developed to create confidence by letting the physician ask it several types of questions (see Fig. 10).

MYCIN frequently asks questions that seem a little baffling to the physician; they just don't seem to be relevant to the task at hand. So he or she might ask why that question was asked, and MYCIN would explain that it's trying to evaluate a particular rule. If the physician types "Why?" again, meaning "Why are you trying to evaluate that rule?", MYCIN replies that it's trying to evaluate that rule to establish the truth of a condition in a higher level rule. By asking why repeatedly you can work right back up the chain to what the ultimate goal of the consultation is.

After the program has made its recommendations, one may ask "How (did you arrive at that conclusion)?" and MYCIN will explain its line of reasoning in terms of the rules that were invoked. MYCIN can even handle a question of the form, "Why didn't you consider something else?" as shown in Fig. 11. The rules in MYCIN have a simple format, which makes it relatively easy to translate and display them in fairly standard English.

MYCIN also was significant in that it gave rise to a whole family of other systems, as shown in Fig. 12. EMYCIN is really not an expert system in itself, but is just the inference part of MYCIN, i.e., MYCIN without the medical knowledge. Because of that separation, one can add new knowledge in related domains or totally different domains and produce different expert systems. And that's what has happened over and over in the past few years. This family tree continues to grow and spread beyond what's shown in the figure. For example, Texas Instruments sells a product called "The Personal Consultant" which is a direct descendant (if not a twin sister) of EMYCIN.

How does MYCIN create confidence in the user?

- Answering "Why?" (Why did you ask me that?)
- Answering "How?" (How did you arrive at that conclusion?)
- Answering "Why not X?" (Why did you not consider X?)

MYCIN's simple rule format and friendly explanations in "English" are the key.

Figure 10. Questions that can be asked of MYCIN.

MYCIN Explanation

User: Why didn't you consider Streptococcus as a possibility for Organism-1?

MYCIN: The following rule could have been used to determine that the identity of Organism-1 was Streptococcus: Rule 33.
But Clause 2 ("the morphology of the organism is Coccus") was already known to be false for Organism-1 so the rule was never tried.

Figure 11. Example of an Explanation in MYCIN.

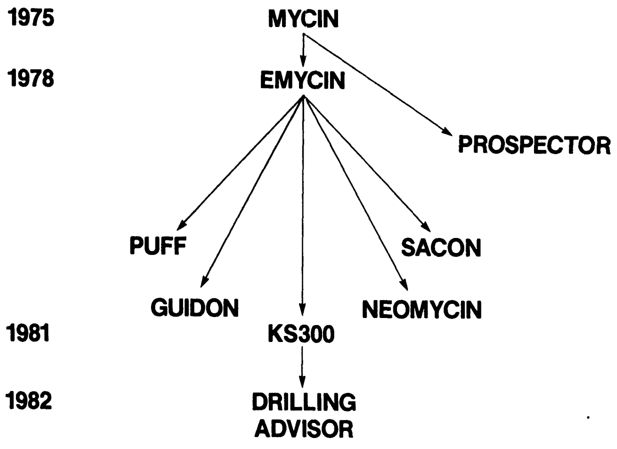

Figure 12. The MYCIN Family Tree.

I won't have time to talk about all of the systems shown in the figure, but I'll mention a few. PUFF (PULmonary Function diagnosis) analyzes data collected by a pulmonary function machine at Pacific Medical Center in San Francisco. It interprets the data and prints out a report, in standard format. About 80 percent of the time, the report says what the expert interpreter would have written himself, and he just signs his name at the bottom.

GUIDON is a system for teaching students about the medical knowledge that is contained in MYCIN. GUIDON also led to a reformulation of MYCIN, called NEOMYCIN, which made explicit the distinction between the domain knowledge (the facts, rules, relations, etc.) and the problem-solving strategy knowledge, and that became the basis of other intelligent tutoring systems.

Let me discuss one more expert system with which I was involved. This application took EMYCIN completely outside the area of medicine and into the domain of structural engineering. By adding knowledge about structural analysis, we created a small prototype system called SACON (Structural Analysis CONsultant). SACON advises engineers on the use of a large structural analysis program called MARC. The advice consists of a high level analysis strategy to use along with a list of specific recommendations. SACON was developed in a very short time, less than six months, and was basically a demonstration that this technology could work in other areas as well.

Figure 13 shows how SACON would look to the user after he or she answers questions about the type of structure (shape, material composition) and the type of analysis desired (material behavior, accuracy, etc.). SACON concludes its consultation with two kinds of recommendations. One is something called the "analysis class". In this case the recommended analysis class is "general inelastic", which was one out of about 36 different kinds of analysis classes that he could choose from. The second kind is a list of specific recommendations about certain input parameters that are provided for the program.

Since this was a concept demonstration, we didn't actually go as far as to generate the actual input data to the MARC program in the form of a tape or punch cards or whatever, but that could have been done straightforwardly. (I know of at least one other similar sort of expert system, a consultant for a seismic data analysis system, that does generate the input data.)

Figure 14 shows a typical rule in SACON, having to do with the type of material in the substructure, the non-dimensional stresses which are calculated by the system, loading cycles and so forth, and then concluding that fatigue must be considered as one of the stress behaviors.

The key ideas that came out of SACON were (a) complex computer systems require consultants to help non-expert users, and (2) an expert system can be an effective replacement for computer manuals and local "wizards." Perhaps this is an appropriate place to allay the fear that these expert systems are putting people out of work. In the case of SACON, there were a very small number of wizards who thoroughly understood the MARC program. They spent a lot of time on the phone talking to customers instead of doing the R&D work that was their primary responsibility. For them, an automated consultant was a way to save them time and let them focus on more productive tasks. This situation is typical.

How SACON looks to the user:

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:

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

Figure 13. Example of SACON Output.

SACON RULE

IF: The material compcsing the substructure is one of metal,
The analysis error (in %) that is tolerable is between 5 and 30,
The non-dimensional stress of the substructure is greater than .9, and
The number of cycles the loading is to be applied is between 1000 and 10,000,

THEN: It is definite that fatigue is one of the stress behavior phenomena in the substructure.

Figure 14. Example SACON Rule.

OPPORTUNITIES FOR NDE

Based on analogy with other areas of science and engineering, I can see a number of different opportunities to use expert systems technology in NDE.

Data Interpretation

A typical problem in this category is, given the location and size of a defect, the stress to which the piece is to be subject, and perhaps some other physical and/or environmental data; is the defect serious? This is clearly an area where judgmental, experiential knowledge plays a major role.

Signal Characterization

Everyone knows that measuring devices can produce erroneous data. It is common, for example, to detect artifacts in signals, induced by the geometry of the object under test or by the environment. Signal peaks may be spurious, transducers can generate false signals, etc. Each source of data has a relative likelihood of credibility. Characterizing signals requires the application of a large number of heuristics.

Control of Data Collection

Intelligent control of how data are collected can often compensate for collecting large quantities of data which must be stored and later culled through at great cost in time. For example, one might use long wavelengths for routine scanning, switching to shorter wavelengths only where closer inspection is required because of a higher load in that area, or detection of an unusual signal, or because of historical data, etc.

Integrated Sources of Knowledge

Defects in a material part sometimes can be corrected on the manufacturing line by experienced line operators who seem to have an almost magic ability to twiddle the right knobs and correct the problem. For problems that are new or especially complex (e.g., when the "disease" is due to the interaction of multiple failures), the analytical skill of well trained engineers or scientists is required. Knowledge based systems can integrate multiple sources of knowledge and achieve better problem solving performance than any one source of expertise. This has been demonstrated many times, but a classic case is the expert system called PROSPECTOR, which combines the knowledge of several expert geologists, each of whom is a specialist in one or more models of ore deposits. In 1980, PROSPECTOR became the first expert system to achieve a major commercial success in predicting a large molybdenum deposit in Washington state.

Inspection Plan Designer

Given a new part to be fabricated, how should it be inspected? Where should you look for the most likely occurrence of defects? How do you totally cover the volume of interest, which involves selecting angles, the appropriate transducer (shear waves vs longitudinal waves, e.g.)? How do you quantify the size and type of a defect once it's located? The seriousness of a defect depends on the applied load--a smooth sphere inclusion is not as serious as a defect near a sharp edge, which can crack. One may need two or more transducers in a pitchcatch relationship, or one may need a different technique, e.g., x-rays instead of or in addition to ultrasonics. Clearly, there are numerous issues that must be considered, and experiential knowledge, rather than textbook procedures, can make the difference between an excellent and a mediocre inspection plan.

Intelligent Front End to Simulation Models

Quantitative simulations of the behavior of structures under various loading conditions provide critical information to NDE analysts. If your work is like many other areas of engineering analysis, you now have simulation packages that are at least a decade old, contain hundreds of options that must be specified before submitting a run. Moreover, the program itself contains tens of thousands of lines of code, the original authors have disappeared, no one understands why or how the program works in its entirety, and the documentation is both too cumbersome to read and understand, out of date, and doesn't tell you what you really want to know. Each organization has a small number of wizards to whom you go in order to learn how to set up a simulation that applies to your specific problem.

This is an excellent area for using expert systems to help nonexperts, and there have been a number of successful demonstrations of intelligent front ends to complex software packages. I've had direct experience with such systems in the areas of structural analysis and seismic data processing. The IFE can conduct a dialogue with the user, querying him or her about the specific application in the familiar terms of the domain. The conclusions drawn by the IFE are then translated into data processing terms -- what values go in which fields of which records. The analyst no longer need be a data processing specialist.

Expert System - Simulation Hybrid

Carrying the idea of combining a reasoning program with a numerical simulation one step further, imagine a closed-loop system where an expert system generates a hypothesis about the type and location of a defect, sets up and runs a simulation program to explore the consequences of that hypothesis, and finally interprets and uses the results of the simulation to generate alternate hypotheses.

These are just speculations. I'm certainly a novice in your field and I'm just going on the basis of experience of similar applications in other fields.

SELECTING AN APPROPRIATE APPLICATION

Here are some guidelines in selecting an appropriate application for applying this technology:

1. The solution to the problem primarily involves symbolic, not numerical reasoning. If there's a way to do it algorithmically using a numerical method, you should do it that way. Don't rely on expert systems.

2. Since we are talking about expert systems, there has to be an expert. I think that's been overlooked, that somehow, these expert systems will magically produce expert performance. They are based on knowledge obtained from experts.

3. It has to be a problem where somebody cares about the solution. The eventual fielded system has to either save people time or money or increase productivity.

4. The problem needs to be bounded in scope or you'll never finish. We have some of our own heuristics about that. If a human expert practitioner can solve the problem in a few minutes, it's too easy. If it takes the expert a week or more it's likely to be too difficult. So a problem that takes an expert a few hours to a few days to solve is appropriately bounded.

5. There should be a vocabulary of the field which is not unbounded. There should be no more than a few hundred terms that would completely describe all the objects and relationships that exist in the area.

6. The experts should agree with one another on the solutions to the problems. When the experts disagree among themselves, there will be a major problem in establishing the credibility of the system you develop.

7. Numerous test cases should be available. Experts generally cannot tell you what they know directly, but can impart that knowledge indirectly by solving lots of problems. Thus, at present, expert systems are developed by analyzing cases.

8. Combinatorial problems are often likely candidates. That's one area where it's difficult for humans to perform very well. We are generally poor enumerators when there are a large number of possible solutions to a problem, whereas machines are very methodical about that.

9. Finally, you ought to be able to show progress in an incremental way so that your sponsor doesn't have to wait for two or three years before anything comes out the other end of the tunnel.

CRITICISMS

Critics of expert systems have frequently pointed out that the knowledge in them is shallow, that they don't exhibit any common sense, and that some problems that would be obvious to you can't be solved by the expert system because it only knows what's in its knowledge base; if it requires using some piece of knowledge that's not in it, it fails. And finally, it doesn't improve with experience. You run the same system day after day after day and it solves the same problems but it doesn't get any smarter.

These are all the areas that we are aware of and they are all areas of research. Rather than say that we have been overselling the systems, I would say that they are not oversold; they are just merely underdeveloped.

LOOKING AHEAD

I'd like to conclude with some personal remarks on where I believe the field is heading over the next 5 to 10 years.

Knowledge Acquisition

Knowledge acquisition is a well-known bottleneck in the development of practical expert systems that require very large knowledge bases. There are two major problems here. One is the time-consuming process of entering thousands of chunks of knowledge into a knowledge base. At present we do this more or less manually, one chunk at a time. Moreover, we have to run the evolving system through hundreds to thousands of cases in order to verify the knowledge acquired thus far and to see what's missing. The second problem is that knowledge bases always have a structure to them; that is, the objects in the knowledge base have particular interrelationships that are important to solving the problem, or at least to solving the problem efficiently. Eliciting that structure is a design activity that requires currently creativity and a lot of handcrafting. That's why we now have to rely on experienced (and therefore scarce) knowledge engineers to build knowledge systems. Thus every new system is built from scratch, and there's no economy of scale.

I think the research now going on at Stanford and other places will eventually widen, if not totally break, this bottleneck. Meta-DENDRAL, now ten years old, demonstrated that one can acquire new knowledge from examples. More recently, one of our Ph.D. students developed a program, called RL, that uses a rough mode, or half-order theory, of the domain in order to guide a systematic search through a space of plausible concept definitions and associations. Preliminary results show because of the rules and metarules learned. Another mode of learning, learning by analogy, is a promising area for constructing KBs. To be successful here, we have to learn how to find the appropriate analogies and to use the analogies correctly. I wouldn't hold my breath waiting for breakthroughs in this area, but I'd keep my eye on the work of Doug Lenat and his colleagues at MCC. I refer you to his recent article on this topic in the Winter, 1986 issue of AI Magazine. Lenat is embarked on a ten-year project to build an encyclopedic knowledge base, structured so that one can search efficiently for analogies and exploit them when direct knowledge of a particular domain is incomplete.

Multiprocessor Architectures

I believe we will see an increasing amount of activity in both the hardware and the software areas, aimed at speeding up the execution of large expert systems. On the hardware side, it's clear that we continue to move in the direction of higher performance and lower cost uniprocessors, and it's also pretty clear that we will see more and more examples of multiprocessor machines with multi-instruction, multidatapath architectures on the market over the next decade. The real challenge is discovering how to program these beasts to fully realize their potential for parallel symbolic computation. My own feeling is that we're going to have to take the low road on this for a while, looking at specific applications, or classes of applications, finding the parallel problem solving tasks that are inherent in that domain, appropriate language in which to implement the system and then a specification for a machine architecture that can support the whole activity -- in other words, the multi-system level approach we're taking on our own advanced architectures project. Only after a few successes will we be able to step back and see what generic progress we've made. I don't see any breakthroughs here for at least another few years, but I think we can see an order of magnitude speedup very soon simply from using several tens of processors and reasonably intelligent techniques for knowledge and data base management. In five years I expect to see two orders of magnitude speedup, and in ten years at least three (but that's based only on faith that a few geniuses will emerge to crack this problem).

Workstations and Distributed, Interacting Systems

Powerful workstations for symbolic computation are rapidly getting down to a price where we can start thinking about them like we now think about terminals, in other words an ordinary piece of office equipment that you might put on every worker's desk. When this happens, I think we'll see a quantum jump in the level of work performed by executives, managers, administrators and secretarial personnel. High-resolution, multiwindow workstations will be as commonplace as the telephone (and in fact the telephone may be just one component of it). Having N times as much processing power as we now have in our centralized time-shared systems will have a payoff not so much in the quantity of computer use as in the quality. The key will be in the sophistication of the user interface that will be permitted by the enhanced processing power of the workstation. Even we executives will be able to understand how to retrieve a vital piece of information that we can't describe exactly, or get help with a planning or budgeting task, or make marginal notes on an on-line document, without having to ask someone for help or read a manual, because our workstation can represent and utilize a model of what we know and don't know, and draw inferences from our often ambiguous commands. After all, the Macintosh can trace its roots to some AI ideas in the early 70s. More recent work on Intelligent Agent, Knowledge-based programming and in language understanding is likely to lead to these simple-to-use, intelligent job aids --"White Collar Robotics" as Business Week described the field in its February 10th issue.

REFERENCE

1. B. G. Buchanan and E. H. Shortliffe, Rule Based Expert Systems, (Addison-Wesley, Reading, MA, 1984).

DISCUSSION

Bill Karp, Westinghouse: I would like to make a comment and pose a question. At Westinghouse, we have been in the business of artificial intelligence as it impacts the field of NDE. The comments that you made with regard to paying attention to just exactly how you define and identify your expert application in fact is absolutely critical and very often is not a specific individual.

Just to give you a feel for where I'm coming from, we developed some automated systems for the interpretation of eddy current data. Our experts were, in fact, our scientists and engineers that put the concepts together. To get this thing really to work, we found out near the end of the program that if we had taken advantage of the experts in the field, specifically in the field of the inspectors, right up front in the planning of the program, things would have gone much more smoothly. So, identification of expert or the experts is absolutely critical.

R. Engelmore: Yes, it really is and they have to be identified early. Also, it requires quite a bit of time for the expert to participate in such a project and that can be a real problem. Experts are, by their nature, scarce. They are always needed elsewhere.

We found that you usually can't convince an expert's boss to release him for a task like this for much time, so you have to jump over his boss and go to his boss's boss who has a little broader or longer-term view of the goals of the organization and is willing to tell the expert's boss to make this person available.

Bob Green, Johns Hopkins: Since even experts or wizards make mistakes, what consideration is given to false positives or redundancy in these programs? And can we expect malpractice suits against computers?

R. Engelmore: It's a good question that comes up all the time. Yes, these expert systems are going to make mistakes because they are based on the knowledge of humans, who make mistakes. One should not necessarily expect any better performance than you are going to get from humans.

I'd like to think, at least for medical system applications and perhaps some other critical ones, that these expert systems will be regarded as providers of second opinions. It will still be the responsibility of the primary physician or the primary user of that system to make the final decision. I think that's the only way it's going to make sense for a while.

The other side of the coin is (again, with analogies to medicine), when these kinds of systems which are shown to perform very well are available out in the field, will there be malpractice suits for people who <u>don't</u> use them?

Dick Berry, Lockheed: One of the things that we are seeing more and more is that the military is interested in getting automatic accept-reject systems built-in for many of the routine inspections that are carried out in the military in ammunition and a variety of other things. Has AI made any inroads in helping to solve this problem?

R. Engelmore: I'm not aware of any inspection systems, using AI, that are actually in use today, but there may very well be some prototype efforts going on now, and there may be people in this room who could answer that question.

Ward Rummel, Martin Marietta: We have one at Martin.

Steve Huber, NADC: When you mentioned about the "no improvement" experience, it comes to mind immediately: Why isn't the feedback built in? For example, if a doctor makes a diagnosis and the patient dies, he tells the system, "The patient died," and therefore the system would refine its statistics. Has that been done? And if it has been done, why has it failed so far?

R. Engelmore: There actually have been some attempts to do just that. There's a program at Stanford called "RX", which looks over a large medical data base and tries to make some interesting statistical correlations that people have not made before, thereby learning new knowledge from past experience. So that's one attempt. But there haven't been very many such efforts to do that.

Now the whole idea of machine learning is a big research topic, getting programs to actually develop knowledge bases, either learning from examples or learning from textbooks, or learning by sort of looking over the shoulder of the practicing expert as he solves the problem, and keeping records of what he or she does. That's a research area now, and our lab for one is actively involved in it.