MACHINES AS EXPERTS

Introduction 1

Artificial Intelligence and Computation
Expert Systems
Knowledge

The Expert Systems of the 1970s

INTERNIST
VM
EMYCIN
PUFF
SECS

The Expert Systems of the 1980s

MYCIN
SACON
GUIDON
MOLGEN
Offshoot Expert Systems

Expert Systems Research of 1980s

Hand-held Expert Systems
Intelligent Assistants
DART: Diagnosis of Computer System Malfunction
VLSI Design Aids Project

Expert Systems: Where Are We Going in the 1980s

New Fields of Application: Home Entertainment and Advice-giving
"Intelligent Agents" for Computer-based Systems
Signal Understanding Applications
Scientific Developments of Critical Importance (Breakthroughs)
Hardware
The Knowledge Engineering Industry
Where are the Knowledge Engineers
The Future Need and Promise

Epilogue: A New Frontier of Knowledge

Acknowledgments

References

INFORMATION TECHNOLOGY

Unprecedented Opportunity

PREFACE - John Pascoe

KNOWLEDGE-BASED REASONING SYSTEMS

Chapter 1. The calculus of reasoning - John Shepherdson
Chapter 2. Logical reasoning in machines - J A Robinson
Chapter 3. Machines as experts - Edward Feigenbaum
Chapter 4. A prototype knowledge refinery - Donald Michie
Chapter 5. Seeing machines - Michael Duff
Chapter 6. Walking machines - Robert McGhee
Chapter 7. Computers in medical bio-technology - Walter Bodmer
To introduce the topic of where expert systems activity is going in the 1980s, I will examine somewhat discursively some historical themes and background ideas.

Artificial Intelligence and Computation

The American Association for Computing Machinery, when it gives its major scientific prize, gives the prize as the Turing Award. We should all keep in mind, that the British, with Babbage and Turing, started modern computation and artificial intelligence.

Turing's view of a computer was much more general than the view of computing that is commonly held. The usual view is that the computer is a calculator of numbers, that in the memory sit entities encoding binary numbers manipulated by arithmetic units producing calculations. That is the narrowest possible view of what a computer is. Turing in fact never held that view. His view was that the computer was a general symbol-manipulating device capable of conducting the manipulation of symbols by any well-defined process. It is that view of computing that motivates the field of artificial intelligence (AI) and that is applied in our knowledge-engineering expert systems development.

Expert Systems. An expert system is an intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. The knowledge necessary to perform at such a level, plus the inference procedures used, can be thought of as a model of the expertise of the best practitioners of that field.

In an expert system the inference procedure is what is used to draw conclusions from the facts and heuristics -- i.e., from the knowledge in the knowledge base -- to solve problems, offer advice, give diagnoses -- whatever the tasks are. The inference procedure works over the knowledge in the knowledge base. Some of us in the AI field specialise in inference procedures. We are world experts on the construction of inference procedures, and Alan Robinson discusses this in his paper about reasoning by machine.

Knowledge. Donald Michie and I often use the term 'knowledge engineering' and it behooves us to say something about what we mean by
'knowledge'. When I refer to 'knowledge', of course I mean symbolic knowledge and I consider two types of knowledge. One type I refer to as the 'facts' of a domain of discourse. Facts are things that are generally agreed upon by the experts, they are things that professors will lecture about, they will write about, scientific journals will publish them, and one can obtain consensus about them among people in the field.

The facts of an area of work do not constitute the main body of knowledge in that area of work, as we all know. All professionals know that most of the information necessary for high levels of performance in an area are not the factual things you read about in the books but the experimental things, the things you learn on the job, the things you learn as an apprentice, namely, the rules of good judgment. George Polya, the great mathematician at Stanford -- still going strong at 93 -- for a while wrote books under the title Patterns of Plausible Inference. The popular version was How to Solve It. They are marvelous books. He discusses heuristic reasoning, 'the art of good guessing'. This is the essence of expertise. The world is too complicated a place for us to know definitely what to do as we try to solve problems. So we come equipped as we are with our expertise; we come equipped with rules of thumb called heuristics -- non-rigorous rules that allow us to cut through the complexities of problems to get through to answers that are good enough most of the time.

Now, that sort of knowledge, the heuristic knowledge, is generally private knowledge of experts. It is not taught in textbooks; it is not given in lectures. You acquire it through internship, through PhD programs, through apprenticeships. You work at the feet of the master and gradually by osmosis you acquire the expertise necessary to perform at a high level.

I shall mean both facts and heuristics when I speak about knowledge. And I shall speak about knowledge held in data structures called 'knowledge bases' to distinguish these from data bases. Data are something that you use: data confront you when you have a problem. Knowledge is what you apply to solve that problem. Knowledge persists over a fairly long period of time. You go to a university and you manage to keep up with your field in order to acquire the proper knowledge to process the data you need each time a new problem comes in -- the data you have in front of you.

Let me try to summarize what the knowledge engineer's task is. The knowledge engineer comes out of AI research, which is part of computer science research. The knowledge engineer comes equipped with understanding of methods of representing knowledge symbolically in the memory of the machine and of methods of symbolic inference. He works with experts in some particular field of the world's work, which I call 'the problem domain' in order to extract from those experts the facts and heuristics, the expertise, that constitute expert knowledge in that field. He combines both fact and heuristics to produce programs that do a variety of problem-solving tasks, form hypotheses from data, propose solutions to problems, offer advice to physicians, offer therapy, and so on.

The parent science of artificial intelligence itself faces three critical scientific problems. The first is the knowledge representation problem. Given that you have expert knowledge, how do you represent it effectively in the memory of a computer so that it can be used to solve problems? Second, is the design of the inference procedure itself, which is a problem of knowledge utilization. What are the most effective methods for inferring things from the knowledge? The third, which is the most critical problem, and I'll come back to it, is the problem of knowledge acquisition. How do you get into the memory of the machine the knowledge that makes up the expertise, particularly the deeply buried knowledge, the knowledge that the expert uses but doesn’t know he has? That's the critical, bottleneck problem of the field.
The Expert Systems of the 1970s. In our own fields of specialization, we all understand how long it takes to acquire the knowledge and experience to perform well -- that when we perform we are more 'knowledgeable' than 'smart'. DENDRAL (Buchanan and Feigenbaum, 1978), MACSYMA (Moses, 1975), and MYCIN (Shortliffe, 1976) were key programs in making knowledge the focus -- setting the dominant paradigm for AI efforts in the 1970s. In the following, I give examples of some other programs from the first decade of expert systems, but only in brief description. Most of these will be drawn from the work of my own group, the Stanford University Heuristic Programming Project.

INTERNIST. The INTERNIST program is probably the most knowledge-intensive expert system. Developed by Popie and Myers (Popie, 1977), INTERNIST performs the task of differential diagnosis in the broad field of internal medicine. Its knowledge base encompasses almost 500 diseases and more than 3,000 manifestations of disease.

Because INTERNIST is intended to serve a consulting role in medical diagnosis, it has been challenged with a wide variety of difficult clinical problems: cases published in the medical journals and other interesting and unusual problems arising in the local teaching hospitals. In the great majority of these test cases, the problem-formation strategy of INTERNIST has proven to be effective in sorting out the pieces of the medical puzzle and coming to a correct diagnosis involving in some cases as many as a dozen disease entities.

Although this consultative program is designed primarily to aid skilled internists in complicated medical problems, the program may have a spin-off as a diagnostic and triage aid to physicians' assistants, rural health clinics, military medicine and space travel.

VM (Ventilator Management). A ventilator is a piece of medical equipment that assists a critically ill patient with his breathing. The task of the VM program is to provide real-time advice to clinicians about patients undergoing mechanical ventilation (see Fagan, 1980). For VM, the ultimate therapeutic goal is to remove the patient from the ventilator, and VM gives advice on ventilator adjustments that expedite this goal. VM use is being explored at the intensive care unit of the Pacific Medical Center in San Francisco, which has an on-line event-based patient-monitoring system that automatically obtains measurements of approximately 30 physiological parameters. VM is intended as an extension of this system that will (a) provide a summary of the patient status that can easily be understood by the clinician, (b) recognize untoward events in the patient/machine systems and provide suggestions for corrective action, (c) give advice on adjustment of the mechanical ventilator based on an assessment of the patient status and therapeutic goals, (d) detect possible measurement errors, and (e) maintain a set of patient-specific exceptions and goals for future evaluations.

One interesting aspect of the VM program is that it works with multiple streams of data sent by the monitoring system over time. Monitoring these data requires a time-based perspective of the patient's status. For example, a recommendation that a patient is ready for a change in therapy is made after VM has analyzed both current and recent past physiological statuses.

EMYCIN. EMYCIN (essential MYCIN) provides a framework for building consultation programs in various domains (see van Melle, 1979). It uses the domain-independent components of the MYCIN system (described later), notably the production rule mechanism and backward-chaining control structure. Then, for each particular consultation domain, the system building supplies the rules and parameters of that domain to produce a functioning program.

EMYCIN is not, in itself, an expert system, but a software tool for builders of expert systems. It provides a useful environment with
emphasis on speeding the acquisition and debugging of the knowledge of the new domain.

To date, EMYCIN has been successfully applied in the domains of pulmonary function (PUFF; see below), structural analysis SCON; (see Bennett and Engelmore, 1979), and oil well drilling. A skilled user of EMYCIN can, with the help of a domain expert, program a small consultation system very quickly, often in less than one week.

PUFF (Pulmonary Function Disease Diagnosis). PUFF (see Osborn et al., 1979), a computer program for the interpretation of standard laboratory measures of pulmonary function, was implemented in EMYCIN. About 50 quantitative parameters are calculated from measurement of lung volumes, flow rates, and diffusion capacity. PUFF uses these measurements, as well as the patient's history, the diagnosis of the referring physician, and its knowledge of three types of pulmonary diseases, to produce a report and diagnosis of the patient's condition. The PUFF system is now in routine use at the Pacific Medical Center. Eighty-five percent of its reports are signed, unaltered, by the physician, after review.

SECS (Simulation and Evaluation of Chemical Synthesis). The development of new drugs and the study of how drug structure is related to biological activity depend upon the chemist's ability to synthesize new molecules as well as his ability to modify existing structures, e.g., incorporating isotopic labels of other substituents into biomolecular substrates. The SECS program assists the synthetic chemist in designing stereospecific syntheses of biologically important molecules (see Andose et al., 1979).

A spin-off of SECS, XENO has been programmed to predict the plausible metabolites of a given xenobiotic so that they may be analyzed for possible carcinogenicity.

A META-SECS top-level plan generator has been outlined to reason using synthetic principles and conclude plans that will then be used to guide the existing SECS program in synthetic analysis. The first-order predicate calculus is being used to represent the synthetic strategies, and an inference processor is currently in design stages.

The programs of the SECS Project have achieved substantial use among chemists in drug and chemical industries and in university laboratories.

The Expert Systems of the 1980s.

MYCIN. Now I would like to discuss one program in more depth, one that Donald Michie has discussed in the context of his knowledge refinery concept, namely, the MYCIN system. It was developed by one of our graduate students, Ted Shortliffe, who was taking a PhD in computer science at the same time he was doing an MD at Stanford. We have a program that allows this at the Stanford Medical School, called the Medical Science Training Program. Shortliffe represented in his person the bridge between the two aspects of knowledge engineering: He was not only a member of the AI research community, but also a member of the problem domain community himself. A brilliant piece of work, MYCIN is a program that conducts consultations with physicians about patients' infections. It has expertise concerning infections of two kinds: meningitis infections and blood infections that are bacteremias. (This range could be expanded.) MYCIN conducts an interactive dialogue in a stylized form of English with a physician, leading to a diagnosis of the infecting organisms and therapy recommendations in terms of the antibiotics that need to be given.

MYCIN has a particular inference procedure, a very simple one, called backward chaining. This is no more than working backwards, as a mathematician would do from a theorem to be proved, to known facts or
laboratory results that may be available. So MYCIN, in using that procedure, constructs a line of reasoning from possible offending organisms back to known patient data and laboratory material. Since it is constructing a line of reasoning, it can explain that line of reasoning to the physician.

[Figure 1]

Figure 1 shows a 'chunk' of knowledge which happens to be Piece No. 85 in the MYCIN knowledge base. MYCIN has about 500 rules of this sort. A rule consists of an IF side and a THEN side. The IF side contains the relevancy conditions for this rule, sometimes called the invoking conditions or the pre-conditions for this rule. If those things are true, THEN some conclusion follows. (In this case, there is a lot of medical jargon it is not necessary to read.) That form of medical English is the form in which the expert physicians build the knowledge base and it is the form in which the physicians using MYCIN see the knowledge. Underlying this is a computer language called LISP, which the user and the expert never need to see.

The rule of Figure 1 will be evoked from the memory and used, possibly in a line of reasoning. The conclusion that follows from those conditions is that it is possible in this case that the identity of the infecting organism is a pseudomonas-aeruginosa. In the rule is a number .6, which looks like a real number, but isn't really. It is a strength of feeling. When the expert gives us this rule we ask him: "On a scale from 1 to 10 or from .1 to 1.0, how strongly do you feel that that conclusion follows from those conditions?" In this case, the expert said 6 on a scale of 10, which means we call it 'suggestive evidence'. We have a way of accumulating those feelings into what we call a 'cumulative believability factor' -- accumulating the inexactness experts feel about a field of knowledge. Something like 9 would mean, "I really think I understand this," and something like .2 would mean, "I hardly know what I'm talking about, but it's better than nothing."

We can also see in the figure the diagnosis MYCIN has made. If the doctor does not like this, he can look at the second recommendation, the third recommendation, and so on. Having advice like this is important to physicians because the issuance of broad-spectrum antibiotics indiscriminately is one of the great medical problems. It is much better to target the antibiotics carefully. Most primary-care physicians are not sufficiently up to day to be able to do that. They need to act quickly, yet avoid malpractice, and they therefore give known broad-spectrum antibiotics. So MYCIN is a very useful system.

Next in the figure is an example of evoking the line of reasoning. The user asks the question in the form: "Why didn't you consider streptococcus as a possibility for Organism 17?" MYCIN's adjunct system, TEIRESIAS, responded: "The following rule could have been used to determine that the identity was streptococcus, Rule 33, but Clause 2 [and then it quotes it: "The morphology of the organism is streptococcus"] was already known to be false for Organism 1, so the rule was never tried." By "already known to be false" it could be, for example, that in the interactive dialogue the primary-care physician said that the laboratory test indicated that the organism was a rod, not a coccus. That was already known to be false, and so the rule was never tried; it did not consider streptococcus as a possibility for Organism 1.

In other words, if MYCIN is building a line of reasoning, it should be able to explain it to the user. We regard it as an engineering essential that the programs do this. Otherwise, a physician, a military commander, or an engineer will never be willing to trust such a system. You cannot build credibility in a system that is strictly a black box. You have to build a totally transparent system whose reasoning processes are visible and self-evident to the customer.

How good is MYCIN? An evaluation was done on the judgments of MYCIN
on a variety of meningitis cases versus those of other agents who were performing those judgments: faculty members at Stanford, faculty members at other places, infectious-disease faculty members versus non-infectious-disease faculty members, interns, residents, students, and so on. Judged by a panel of national experts in infectious-disease therapy, MYCIN came out at the top of that list, just about tied for first with the infectious-disease faculty members who helped us build MYCIN at Stanford.

When I gave a talk in Switzerland last summer, my associates and I were able to demonstrate MYCIN over the packet-switched networks to Stanford and it was very exciting. There was a physician in the room -- Marvin Minsky's wife -- who did not know anything about MYCIN. We had her come up and invent a case which we entered into the system; then MYCIN diagnosed the case correctly. The physician was amazed; the audience was amazed; and I was amazed -- not that MYCIN would work but that I could get symbols through on European packet-switched networks to Stanford!

I have said that an expert system consists of a knowledge base and an inference procedure. If you subtract the knowledge base from MYCIN, you get the inference procedure, which we call 'EMYCIN' for Empty MYCIN (described earlier). If you add back to this inference procedure the rules for some other area, you can create another expert system. At Pacific Medical Center we found a very congenial expert who ran a pulmonary function testing laboratory. We created a system called PUFF (also described earlier) that analyzes the spirometer data that comes from breathing in and out and produces well-reasoned interpretations that run normally 2, 3 or 4 paragraphs on the patient record. It is examined by the physician who signs it if he likes it and puts it in the file. The vast majority of outputs are signed by the physician and entered into the patient's record.

SACON. I have an engineering example as well to show that the EMYCIN program is more general than its use in medicine. That engineering application of EMYCIN is called SACON. Its domain, in this case, is a complex software package, a giant finite-element-analysis calculation package done by Marc Analysis Research Corporation on Prime computers. It is a superb package. Indeed, the problem is not its quality but its complexity. The manual for this package is about four inches thick. No customer can find his way through this manual to know what is there to use to solve the problem. A military client came to us and said that the structural engineers doing aerodynamic calculations needed a consultation system like MYCIN to help them find their way through this manual -- to find out what to do, given a problem. SACON is the resulting system. Its task is to consult with the structural engineer about design specifications for a wing structure, an offshore oil derrick, a nuclear reactor pressure vessel, or something like that and to advise on the plan of calculations for using the Marc package. We worked with an expert who had helped develop the Marc package. The Marc Corporation happens to have their headquarters in Palo Alto, so it was very convenient to work with them.

[Figure 2]

Figure 2 shows a piece of knowledge in SACON. It looks very much like MYCIN. In this case, the IF side talks about heavy metals and a certain analysis area that you are dealing with. Item 4 says you have got to load this thing up 1,000 to 10,000 times and, if you do that, it is definite that you ought to calculate something about fatigue in these metals. And that leads the program on to consider other things. This is one of 300 or 400 different SACON rules for advising engineers on the use of the package.

SACON was designed by one graduate student in one academic year, working with an expert from the Marc Corporation under the supervision of one of our research scientists a small portion of the time, so it is a relatively light project. It used the EMYCIN software package as a starting point.
GUIDON. It takes a long time to dig out the knowledge that makes up a given expertise and you might say that, having done so, you'd like to be able to sue it for more than one thing. And the obvious extra thing is teaching. If, in fact, we are eliciting a type of knowledge that has never before been seen explicitly, namely, the application of heuristic knowledge rather than factual knowledge, then it is really worth teaching that knowledge instead of forcing the student to absorb it by osmosis from working with an expert. So, one of our graduate students constructed the GUIDON program which was capable of teaching any knowledge that EMYCIN could apply.

It is not easy to teach an area of expertise. There is good teaching and bad teaching. So GUIDON contains rules of expertise for good teaching and, when merged with the rules of expertise of a domain, you get a very good teacher that poses problems, understands the student's difficulties, and leads the student through tutorial examples to an understanding of problem solving in that area.

Once GUIDON was running, it taught bacteremia diagnosis and therapy, meningitis diagnosis and therapy, structural engineering advice, and pulmonary function diagnosis; it has now, of course, taught many other things that EMYCIN can do.

MOLGEN. I have some ancillary themes to discuss, namely, in molecular genetics. We began this work in 1965 -- by "we" I mean myself and Professor Joshua Lederberg of the Stanford Medical School, the geneticist. But we began, not in genetics, but in organic chemistry. We were doing some work on the analysis of mass spectra of organic molecules, but each year I kept asking Professor Lederberg when we were going to work in his own field. There is a theorem around that "everything can be automated up to what I do," and it tooka 10 years to get Lederberg to agree to do AI work in genetics. In 1975, however, it looked like there was a rare opportunity. About two years earlier, Stanley Cohen and his friend Herb Boyer in San Francisco had invented the fundamental methods of manipulating DNA. There subsequently began an explosion a few doors down the hall from us. So it seemed that if we were going to build a knowledge base in a field of work that was immediately relevant to the world, nothing could be better than capturing this knowledge as it exploded right out of the door of Stan Cohen's laboratory -- with Paul Berg and others close by.

So we began to design a system called MOLGEN (for Molecular Genetics). It had a number of goals. One owes the design of gene-cloning experiments. It turns out that one of the tricks for enticing experts to bother to spend their time sharing their expertise is to provide them with software tools they didn't know could be built. In this case, we are very familiar with techniques for symbol manipulation by computer, whether those techniques are smart or not. So we gave the experts symbol-manipulation techniques for analyzing DNA sequence data. The number of sequences was beginning its exponential climb. Last year, 50,000 new bases were added to the literature. It will be one million this year. So sequence-analysis data are very important. We also gave the experts some programs for tasks that are not complicated for people to do, just extremely tedious and difficult to get right. One of those problems is called restriction site mapping.

We set our sights to replicate the reasoning that went into a rat-insulin gene-cloning experiment. (Figures 3 and 4 show this.)

[Figure 3]

[Figure 4]

Figure 4 shows the output of the program. The program decides it needs to combine the rat-insulin gene with a gene that confers resistance to antibiotics (for reasons that will be apparent later). To do this, it has to put linkers on the end of the rat-insulin gene and then cleave that gene and the resistance gene with restriction enzymes in a certain way. And then it ligates (glues) the genes which
results in a recombined piece of DNA, shown in the Figure as a circle with i for the rat-insulin gene and r for the resistance gene. The program introduces that piece of DNA into a bacterium (some bacteria get it, some don't). An antibiotic is used to screen them, because only those bacteria that contain the rat insulin gene also contain the resistance gene. The others will die. A pure culture remains, and that pure culture will then produce rat insulin.

Such is the chain of reasoning that the program went through -- a fairly sophisticated piece of reasoning. Of course, it did not reason through it the way I have just explained it. It reasoned through it mostly working backwards. That should not come as a surprise to anyone who reasons about anything. Usually if you have a goal, you reason backwards from the goal to known conditions and things you can pull off the shelf.

The first thing that happened to the MOLGEN program was that word about it got around quickly. Within six months we had 200 academic users calling in on our computer network to use programs at Stanford on our guest accounts. The executive committee of the resource then decided it had to limit this activity, otherwise AI research would go down the drain in the service of molecular biology. So we limited it to no more than two simultaneous jobs. The membership of that group still rose from 200 to 300 over the next three months, a vigorous community called Genet. Walter Bodmer's group used it over networks from England; now they have the software in London.

There is nothing more effective in welding a scientific community than bringing that community together electronically. They come into the machine to use the software and lo and behold! They find on the electronic bulletin board some message about some new DNA sequence that someone has found, or some query someone had about what kind of restriction enzyme you might use for some task. "If you know the answer, please respond to . . .", and there's an electronic mailbox listed there -- please respond to Jones, or Smith at MIT, something like that. Electronic mail starts to flow, and before you know it, you've welded a scientific community via the electronic communications means, even though those people are not geographically in the same place, but spread all over the country.

The second thing we noticed was that in the early phases of this guest account business we had a lot of customers, or I should say guests, from emerging genetic engineering firms. The heaviest user of all was a firm in San Francisco that was working on the cloning of the interferon gene. That led to certain problems because both the U.S. Government and the British Government would say, "You don't let commercial firms in on a government-sponsored resource for academic research." That rule is applied rather strictly, except for trial purposes; once in production mode, users should pay their own way.

Unfortunately, there is no easy transition. Someone has to be responsible for making the code move to some commercial system and for worrying about the users' complaints and all the routine things of being in the business, as opposed to helping your friends. So four of us formed a little company called Intelligenetics and started to sell this service on a commercial system under license from Stanford University, which owns the code. This particular series of software entities is now available commercially in a nice, comfortable, and, we hope eventually, lucrative business.

I have already mentioned the effectiveness of the national network in promoting software. We computer scientists have a very powerful means for publishing our results: sending out the code or letting others run it. And it is always easier to bring users to your code than to send your code to them, because you know that your code runs on your machine and there will always be a hundred reasons why it does not run on their machine. Because the national networks make this possible, they are very powerful entities. I was very glad, indeed, to hear about the British University National Network referred to in David
Thomas' paper.

Offshoot Expert Systems. The current trend in expert systems work is to develop software packages. These packages constitute the residue of the experimental work in designing systems and seeing how they run. In essence, this is what we pass on to the next generation of knowledge engineers. If you do a MYCIN you pass on MYCIN -- I should say EMYCIN's clone, to include Donald Michie's AL/X system that is available in Britain. There are many similar software packages. AGE is a system done by Penny Nii. UNITS is the knowledge representation system that was developed for use in the molecular genetics application and it is now used in many other applications. KL1, developed by Bolt, Beranek and Newman, in Cambridge, Massachusetts, is a knowledge representation system. EXPERT is a Fortran-based package developed by Rutgers University. ROSIE is a predicate-calculus-based package that was done at the Rand Corporation in Santa Monica, California; its name stands for Rule Oriented System for Interactive Expertise. OPS is a system developed at Carnegie Mellon University that was used to build an expert system for assisting Digital Equipment Corporation engineers in configuring VAX computer systems. This is not an exhaustive list of all the packages available. But if you do not use one of these packages, you are doing it wrong. You will waste man-years in getting going. These packages embody the expertise of the knowledge engineers themselves as they pass it along to the next generation of people. I might add that, again, in typical Silicon Valley tradition, there are other firms who are organized to market that type of software, one of which is called Teknowledge.

Expert Systems Research of 1980s

Hand-held Expert Systems. What do I see as the areas of work of the 1980s? The profit will be in the hand-held expert system, the hand-held adviser, the hand-held symbolic manipulator; not number calculators, the device into which you plug your knowledge base. It happens that my wife, Penny Nii, is an expert on plant care. Plant care is an interesting case because very few people know how to do it well and you cannot be sued for malpractice! Imagine typing into your hand-held system "my green plant has white spots on the leaves, what do I do?" And it says, "Have you given it fertilizer recently?" And you say, "Yes, I've given it 0-10-10." And it says, "This is ridiculous -- your should have given it 50-9." Another possibility for a hand-held system is one for automobile repairs. The United States is particularly intrigued with "Do It Yourself" activities these days because if you have to pay someone else to do it, you pay that person after-tax dollars and the tax rates are very high. So there is a strong tendency to try to do it yourself and not have to pay someone out of tax dollars.

If the AI people have been faulted for anything over the years, it is for their preoccupation with games. We have seen mediocre game players challenging you in machines with names like the Chess Challenger or Boris. Now imagine some really good ones. Imagine master-level play. Imagine a bridge partner that can do better than anyone you can find on a Saturday night. And so on, and so on, and so on. Those games will be an exciting part of the consumer market.

Intelligent Assistants. As stated in my earlier discussion of the professional market, physicians, engineers, scientists, mass spectrometrists, molecular geneticists -- anyone you can name -- are candidates for a symbolic reasoning assistant; for an intelligent assistant. Something that is not so obvious is what I will call the technician market, which requires aids for the diagnosis of equipment failure and repair of equipment; that is, diagnosis and therapy of sick machines. If we can do it for people even though we do not understand the human body very well, we certainly ought to be able to do it with precision for man-designed machines where we even have blueprints. We have functional specifications. And that is indeed the case.
The thrust in this direction is fueled by the increasingly desperate shortage of trained technical people. You have probably noticed that it is harder and harder to get things fixed. The military finds this particularly provoking because when the United States went to a voluntary military force, the theory was that if you pay people well enough you get adequate brains attached to those people. Unfortunately Congress never decided to pay them well enough. You are getting to the bottom of the barrel, at a time when the devices they are to service are becoming increasingly "high-tech". So the gap is widening between what is needed and what is available in the way of human brainpower to fix those things.

In the military market, I have already mentioned troubleshooting and repair. Another aspect of the military market is "command and control": What is going on now in my battlefield? And planning: What do I do next? Imagine two screens, one of which is reporting all the evidence from the sensors and saying what the merging battle situation is, and the other one has all the resources at the planner's disposal. He cannot keep up with the flow of information nor comprehend how to plan the next step, given the things that are being wiped out in the battlefield -- and so on. So for what is called the "machine-definition" screens, the electronic battlefield, the military commanders believe (and I think they are right) that they need automatic assistance. They need smart agents to help them assess situations and plan. There is a considerable amount of work going on now and through the 1980s in the area, especially by people at the Rand Corporation.

Two examples of expert systems research recently begun, diagnosis of equipment failure and VLSI design, will be important tasks for expert systems in the 1980s.

DART. IBM ran into a very difficult problem when they introduced the 4300 Series. They made a price reduction that was essentially unbelievable for the computer field. This stimulated demand to the extent that there weren't enough field engineers to hire to maintain the 4300 systems -- not the chips themselves, but the systems -- multi-computers linked by telecommunications. Where's the bug? Software? Telecommunications? Interface? Hardware? Finding the bug -- that is the problem we are working on now with IBM, and MIT is working on a similar problem with Digital Equipment Corporation.

The purpose of the new Stanford-IBM project, called DART, is to explore the use of casual, structural, and teleological models of computer systems in diagnosing computer system faults. The research is aimed at developing an adequate "machine-definition" language for encoding such models and identifying general diagnostic techniques. Each model includes a description of the system's "anatomy" and "physiology" and knowledge of how this structure realizes the system's function in terms of general principles of design. The practical goal of the research is an automated diagnostician for a contemporary computer system like the IBM 4331. We also hope to obtain such by-products from the research as a program for assisting field engineers, a training facility for field engineers based on the device models, and feedback on hardware design.

VLSI Design Aids Project. The hottest application area in the 1980s for expert systems is the design for VLSI chips. The last of a series of five articles by Art Robertson in Science last year on the American microelectronics industry essentially said, "Given what we know and where we want to go, you can't get there from here." The limitation was "design smarts": there just are not enough around to cope with the exponentially increasing complexity of chips.

When you try to lay down elements on a chip, you have a certain level of problem. If you double the number of elements on the chip you do not have a problem that is double the size. You have a problem that is increasing geometrically in size as you begin to manipulate the elements. The combinatorics of placing elements down on a chip with
the layer topologies and the geometries of drilling between the layers
gives you a very difficult geometrical, topological, combinatorial
problem.

Experts are great at cutting through such complexities. But there are
two problems. According to Science there are only 1700 experts in the
United States for all the microelectronics firms. Second, as we have
found out, the experts are not all that expert. The field is too new
for it to have developed much expertise. So the work in artificial
intelligence, or expert systems work applied to VLSI, has two aspects.
One is the replication of expertise by dragging out the expertise of
current experts, putting it in programs, and making those programs
available to designers. But the second is the refinement of knowledge
in the sense that Michie writes about. As we look at the expertise
and can discern its inadequacy, we can repair it. Not just we, but we
working with the experts can repair the knowledge, fix it up, tune it,
hone it, make it better, and produce better design methodologies.
That is the real payoff.

The overall objective of the Stanford VLSI design aids project is to
develop intelligent, high-performance computer tools for designing
very large scale integrated circuits. The tools developed in this
project will be integrated into the design automation system that is
being developed by the Stanford Center for Integrated Systems.

The project is currently developing a heuristic layout program (HLP),
which is a technology-independent, knowledge-based program to generate
automatically a geometric (mask-level) layout of an integrated
circuit, given an abstract description of the circuit (e.g., a
hierarchical symbolic layout of the circuit). The program is intended
to serve both as a layout tool and as a test bed for experimentation
with various layout heuristics. HLP follows the hierarchical
planning-with-constraints scheme developed as part of the MOLGEN
project.

Expert Systems: Where Are We Going in the 1980s?

Predictions are risky, and today's prophet is tomorrow's fool.
Nevertheless, here are some of my views on the course of events in the
1980s.

New Fields of Application: Home Entertainment and Advice-giving. AI
scientists have long used games as vehicles for exploring new
concepts. A game generally has a constrained knowledge base and is
highly structured. There is not only a considerable competence within
the AI community for constructing intelligent game programs, but also
a vast consumer market for home entertainment of this sort. In my
view, the money-making potential of this market will make
microcomputer-plus-TV-based home entertainment the dominant market for
expert systems. The concept of home entertainment is broader than
games, however, and include consultation and advice about a broad
range of subjects of interest to the consumer (e.g., financial advice,
garden and plant care). Knowledge bases for these specialities will
be assembled by experts in much the same way that the "how-to" books
traditionally have been compiled.

"Intelligent Agents" for Computer-based Systems. As we and our users
are (jointly and painfully) aware, the use of complex computer
software is a very knowledge-intensive activity. The manuals are too
complex, filled with a mass of detail, easily forgotten; they are
often poorly written; and often contain merely the "facts" about the
system, not the "lore" necessary to facilitate use (the "lore" is
known to the "system-hackers"). In the 1980s, systems will radically
improve the "quality of life" for a user attempting to apply the
software. Such an expert system will have an extensive knowledge base
provided by the system developers, and augmented by the user community
itself, that can be employed to interpret and fulfill user requests.

The manuals of the past will become active and inferential in the
service of user needs. The market for such systems is as broad as the
Signal Understanding Applications. Of the many applications to fields of science and engineering that will be done, one class will be outstanding in its impact -- signal understanding. The AI methodology that underlies expert systems work contains the most powerful techniques known for realizing signal-to-symbol transformations, the essence of signal understanding (as opposed to signal processing). For situations of low signal-to-noise ratios, it makes little sense to use enormous amounts of computation for statistical calculations to tease the little signal from the noise, when most of the understanding can be readily inferred from the symbolic knowledge surrounding the situation.

Scientific Developments of Critical Importance (Breakthroughs). There are many important problems of knowledge representation, utilization, and acquisition that must be solved, but the acquisition problem is the bottleneck problem. An expert system is expert only because it is knowledge-intensive. This problem is now becoming sharply focused, and is beginning to receive the attention it deserves. It is becoming clear that acquisition itself must be a knowledge-driven activity. In the 1980s there will be breakthroughs in two styles of acquisition: first, the acquisition of domain-specific knowledge directly from recorded data (i.e., essentially automatic); and, second, interactive transfer of expertise, in which a program guides and helps the expert in explaining and formalizing his/her knowledge.

Similarly, the technologist/builder of the expert system needs tools to help construct the system. Some, like EMYCIN and AGE (from Stanford), EXPERT (from Rutgers), and ROSIE (from the Rand Corporation), are now available; and the 1980s will see a proliferation of such software packages representing, in effect, "high-level languages" with which to write expert systems. The major inference methods of artificial intelligence will be packed up in this way and will be coupled to powerful representation structures and formalisms.

Hardware. The microelectronic revolution of the 1980's will put the average expert system into a relatively inexpensive and small hardware package. This is an unremarkable prediction based on the progress of VLSI toward million-gate and ten-million-gate chips. The knowledge base will be held in disk, bubble, or semiconductor memories of large sizes, accessed as needed. We will see a spectrum of expert systems, from those of modest scope of knowledge that may be packaged in small, highly portable machines costing less than a thousand dollars to powerful aids for professionals that will reside in work stations costing a few tens of thousands of dollars. Very large knowledge bases will be stored in central repositories, accessed by a communication network as needed.

The Knowledge Engineering Industry. As the pressure builds to apply artificial intelligence to a variety of expert system projects, a new industry (now in its infancy) will emerge. In a fashion analogous to that of the aerospace engineering industry, this new "knowledge engineering" industry will transfer the developments of the research laboratories to the useful expert systems that industry, science, medicine, business and the military will be demanding.

High-technology companies have already begun to build these groups internally (examples: Schlumberger, Ltd; Texas Instruments; IBM). In the United States, new small firms have incorporated, and the best will have a dramatic growth in the 1980s. Expert systems methodology is at the place today that operations research was in the mid-1950s, as it moved from its post-World War II decade of development to its maturity of intense application.

Where are the Knowledge Engineers? Limiting the pace of development of this industry will be the shortage of people -- the new knowledge engineers. Everywhere the shortage of highly trained computer scientists and technologists is being felt. In the United States,
this is currently called "the crisis in experimental computer science." Few universities are training graduate students in expert systems methodology. Because of the heavy non-academic demand, few of the new PhDs are locating in universities, so there is currently no "amplification factor" in the production of trained people. Failure to solve this educational problem will make all predictions about expert systems in the 1980s seem to have been wildly optimistic. Significantly, only in the United States has there been a vigorous discussion of this problem, leading me to the seemingly inescapable conclusion that the United States (moving to solve the problem) will dominate this new industry throughout the 1980s.

The Future Need and Promise. There is a certain inevitability to expert systems research and application. The cost of 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 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, management, the military -- 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.

Epilogue: A New Frontier of Knowledge

Beyond the 1980s, the larger impact on science, technology and society will be the exposure and refinement of the hither-to private heuristic knowledge of the experts of the various fields of practice studied. The ethic of science that calls for the public exposure and criticism of knowledge has traditionally been flawed for want of a methodology to evoke and give form to the heuristic knowledge of scientists. Expert systems methodology is beginning to fill that need. Heuristic knowledge can be elicited, studied, evaluated by peers, and taught to students. Perhaps it is less important that computer programs can be organized to use this knowledge than that the knowledge itself can be organized for the use of the human practitioners of today and tomorrow.

Acknowledgments

Material on the INTERNIST and SEC programs was adapted from the 1980 SUMEX-AIX Annual Report to the National Institutes of Health.

References


Moses, J.A., MACSYMA Primer, Machlab Memo 2, Laboratory for Computer
Science, Massachusetts Institute of Technology, October 1975.


