was 28.5 million years with a confidence level of 99.8 percent.

Perhaps more impressive than the apparent high confidence levels and near identity of calculated periods is the coincidence of the reported times of comet showers and mass extinctions. Alvarez and Muller find the most recent comet shower to have been 13 ± 2 million years ago, the same age reported for the most recent extinction event. The next three or four periodic extinction events fall neatly within the age ranges of the cratering episodes, in part because their permissible age ranges expand with the accumulation of the million-year uncertainty in the cratering periodicity. Errors in the dating of mass extinctions also increase in the more distant past, allowing a doubling up of two extinctions at the seventh and ninth cratering cycles.

Some researchers are not convinced by the statistical arguments for periodicity. Richard Grieve of Brown University, who originally compiled the terrestrial cratering record, is not bothered by the suggestion of spikes of cratering in the record, but periodicity is another matter. "They've done it the best way they can," he says of the analyses, but "it's just not a data set that's amenable to time-series analysis." The problem, he says, is that any million-year date without a crater in the selected record is assumed to lack a crater in reality, and this creates an artificial and misleading background of zero events per million years. Among other problems, Grieve notes that chemical analyses of some of the selected craters indicate an asteroid as the impactor rather than a comet.

One of the few cautionary notes of the workshop was sounded by David Jablonski of the University of Arizona when he noted, on the basis of an admittedly hasty compilation, a curious coincidence between the ages of craters and apparent decreases in sea level. Could the cratering record be biased by a periodic increase in the area of the impactor's target—the continents—due to a fall in sea level? No one had a ready answer.

The suggestion of periodic extinctions by comet showers is stimulating as much constructive hypothesis testing as did the original claim for a devastating impact 65 million years ago, as well it should. The implications are profound not only for the driving forces behind evolution but also for those behind climate change; if periodic comet showers instead of climate change are the ultimate cause of mass extinctions, then presumably comet showers also cause permanent changes in climate.

Tantalizing evidence of the expected association between multiple impacts, climate change, and extinctions was presented at the workshop. Erle Kauffman of the University of Colorado has found apparent climate oscillations in the half million years immediately preceding the extinctions of 91 million years ago. No signs of an impact have yet been found there, but similar oscillations seem to have preceded other mass extinctions, including those now associated at least in part with the impact 65 million years ago. Gerta Keller of the USGS in Menlo Park reported evidence (*3*) of debris from several impacts clustered around 38 million years ago, about the time of major climate changes and extinctions. Expanding on such studies, as well as searches of the sky for the companion, will be the next steps in testing the new hypotheses.—**RICHARD A. KERR**

References

1. J. G. Hills, *Astron. J.* **86**, 1730 (1981).
2. D. M. Raup and J. J. Sepkoski, *Proc. Natl. Acad. Sci. U.S.A.* **81**, 801 (1984).
3. G. Keller, S. D'Hondt, T. L. Vallier, *Science* **221**, 150 (1983).

The Necessity of Knowledge

The essence of intelligence seems to be less a matter of reasoning ability than of knowing a lot about the world

The field of artificial intelligence, or AI, is split into two camps. The "engineers" are trying to get their programs to do smart things, by whatever means they can. The "scientists," a much smaller group, are after a general theory of intelligence, both human and machine.

Either way it is a tough job. Even the enthusiasts have to admit that AI's achievements to date are at best embryonic. The utilitarian approach has produced some reasonably effective software, and in the case of the so-called expert systems that software has begun to be successful in the marketplace (*Science*, 24 February, p. 802). But the main thing that AI researchers have gained on the theoretical front is a certain humility, an appreciation of how awesomely complex the most ordinary human act can be—and of just how much a computer (or a human) has to *know* before it can do much of anything.

On the other hand, it is the computer that gives AI its promise. The fundamental assumption of AI is that the mind can be modeled as a processor of symbols—in effect, as a computer program. Cognition is considered to be a high-level process, which means that it can no more be understood in terms of the firing of individual neurons than a computer program can be understood in terms of the 1's and 0's flitting through an individual memory register.

Given that assumption, the fundamental methodology of AI is strikingly like that of mathematical physics: first turn a set of abstract speculations about the mind into a concrete computer program (write down the equations), and then make that program perform (solve the equations). If it works, then maybe the model was a good one to begin with; if it does not, then maybe a study of how the model breaks down can suggest a better one. At the very least this process enforces a certain clarity and precision, and weeds out ideas that are fuzzy, or incomplete, or wrong.

Historically, AI was a product of the post–World War II ferment in information theory, control theory, and cybernetics; people were writing AI-like programs almost as soon as computers were equipped with enough memory (about 4 kilobytes) to handle them. But it really only emerged as a well-defined field in the mid-1950's. In fact the name "artificial intelligence" itself was only invented in 1956, when John McCarthy, now at Stanford University, used it to describe a summer workshop he was organizing at Dartmouth College.

Those were the days when everything seemed possible. One of the brightest dreams was the creation of a program that would mimic the full range of human problem-solving abilities, from getting

NAME: Primes
STATEMENT
English: Numbers with two divisors
LISP: λ (n) (Apply* (Lisp-Statement Doubleton)
(Apply* (Compiled-Coded-If-Then Divisors-Of) n))
SPECIALIZATIONS: Odd-primes, Small-primes, Pair-primes
GENERALIZATIONS: Positive numbers
IS-A: Class-of-numbers
EXAMPLES:
Extreme-exs: 2,3
Extreme-non-exs: 0,1
Typical-exs: 5,7,11,13,17,19
Typical-non-exs: 34, 100
CONJECTURES:
Good-conjecs: Unique-factorization, Formula-for-d(n)
Good-conjec-units: Times, Divisors-of, Exponentiate, Nos-with-3-divis, Squaring
ANALOGIES: Simple Groups
WORTH: 800
ORIGIN: Application of H2 to Divisors-of
Defined-using: Divisors-of Creation-date: 3/19/76 18:45
HISTORY:
NGoodExamples: 840 NBadExamples: 5000
NGoodConjectures: 3 NBadConjectures: 7

Knowledge in a machine

A frame-like representation of the concept of "primes" [taken from Douglas B. Lenat's program AM].

up in the morning to getting to the airport. In fact, one of the earliest and most influential AI programs ever written was called the General Problem Solver (GPS). Devised in 1957 by Allen Newell and Herbert Simon of Carnegie-Mellon University, and J. Clifford Shaw of the RAND Corporation, GPS was based on the idea that humans reach solutions to problems through a kind of "means-end" analysis: given that one is *here*, and that the goal is over *there*—take steps to reduce the difference.

Alas, the dream tarnished rapidly. In all but the very simplest cases the generalized programs failed, overwhelmed by something called the "combinatoric explosion."

A classic example is the game of chess. As its own well-defined and self-contained world, chess has long been a favorite challenge to AI programmers. However, any chess-playing program quickly has to face the fact that there are some 10^{120} sequences of legal moves in the game. A completely general program would essentially have to examine them all. And yet the fastest computer ever built could spend the lifetime of the universe at it (actually, the lifetime of a great many universes), and still have a long way to go.

The problem turns out to be ubiquitous. Even a much simpler game like checkers has some 10^{40} sequences. So as a purely practical matter, most of the early AI researchers confined themselves to very narrow problems and made liberal use of "heuristics," rules of thumb about the task at hand. In chess, for example, a heuristic might be something along the lines of "IF a pawn has a choice between moving forward or capturing your opponent's queen, THEN take the queen."

Using task-specific heuristics meant giving up any hope of generality—the pawn-capture-queen rule would find limited use in proving theorems—but at least it was a way of pruning the choices down to something manageable.

Now, in a sense no one could really be surprised by this. Humans obviously used specialized knowledge in judging the reliability of a fact, for example, or in deciding whether a solution was reasonable. But the appalling thing was how *much* knowledge was needed. AI systems were already among the largest and most complex programs ever written, and still they needed more. Knowledge was like an addiction. By the mid-1970's the ideal of the general problem-solver had begun to seem hopelessly naïve.

In fact, by the mid-1970's the conventional wisdom in AI had undergone a fundamental change. The essence of intelligence was no longer seen to be reasoning ability alone. More important was having lots of highly specific knowledge about lots of things—a notion inevitably stated as, "Knowledge is power."

Unfortunately, "knowledge" is a very slippery concept, and "knowledge programming" turns out to be a very tangled affair. For more than a decade it has been the greatest single preoccupation of AI research. It is true that the purest and most narrowly defined form of knowledge programming, the so-called "expert systems" that try to mimic the abilities of human experts, have done rather well (page 1281). But that does not change the fact that knowledge programming is in theoretical limbo, much as physics was before Newton.

At the moment, people find themselves grappling with three interrelated issues: the representation of knowledge, which is roughly the machine equivalent of human memory; the control and use of knowledge, which corresponds to human abilities in problem-solving and planning; and the acquisition of knowledge, or what humans call learning.

Representation: It is easy enough to store names and numbers in a computer. But a string of data is not knowledge, any more than the telephone directory is knowledge. Since no one can say what knowledge *is* exactly—30 years of AI have not necessarily improved upon 3000 years of philosophy—AI's only recourse is an operational definition: knowledge is what helps a computer do smart things. It might consist of information about objects and events in the problem domain, for example; it might involve instructions on how to perform in certain situations; occasionally it might even consist of "metaknowledge," which tells the program how to control its own operation.

The oldest and simplest representation scheme is just standard symbolic logic: "All crows are black," and so forth. The advantage is that the statements are precise and well defined. Moreover, logical deductions are easy to automate. Powerful algorithms already exist, and the conclusions are guaranteed correct.

For all its appeal, however, logic programming in its purest form has long since fallen out of favor, and for a familiar reason. A general-purpose deduction algorithm has no way of sorting out essential statements from irrelevancies; set it to work on a knowledge base of any size, and it bogs down in the combinatorics.

(Logic programming has nonetheless retained some passionate adherents, especially in Europe. Moreover, a logic-based language known as PROLOG has been chosen by the Japanese as the programming language of their Fifth Generation project (*Science*, 6 May 1983, p. 581)).

A related problem with logic is that a large knowledge base quickly becomes an amorphous, unstructured listing of statements. Quite aside from being incomprehensible to programmers, such a listing does not reflect the fact that human knowledge is *very* structured—indeed, that human thought is often just a matter of one thing suggesting another.

The current most popular alternative to logic is the *production systems* approach, first developed by Newell and Simon in the mid-1960's. Essentially it is a simplified version of logic program-

ming that allows only certain kinds of statements—an important example being heuristics of the form "IF this is true, THEN do that"—and only certain types of logical deductions. The payoff is that a production system is much easier to handle, and much less subject to the combinatoric explosion. Moreover, they can handle statements about the reliability of the rules.

Also popular are *Frames* and *Scripts*, respectively pioneered in the mid-1970's by Marvin Minsky of the Massachusetts Institute of Technology and Roger Schank of Yale University. These are very elaborate data structures that model human expectations. A DOG frame, for example, would have "slots" for facts that are typically known about dogs, such as BREED, NAME, or OWNER. A RESTAURANT script would have the typical sequence of events at a restaurant, such as being seated, reading the menu, ordering, and so forth.

The list does not stop there—recently, for example, there have been some interesting attempts to codify a qualitative physics or a qualitative psychology—yet there remains a key problem. Any representation of knowledge in a computer is just one rather rigid mapping of the world. But humans are not always rigid. Sometimes, when we are baffled by a problem, we can step back, look at it a different way, and AHA!—the solution is obvious. Whatever is going on in the way of "knowledge representation" inside our heads, it is both high-structured and wonderfully reshapeable. Trying to get a program to do that is one of the foremost goals in AI.

Inference: A perennial issue in AI is the matter of *control*—what does the program do next?

In most AI programs the control knowledge is implicit, buried as heuristics deep in the code. This has the virtue of being efficient and the defect of being rigid; as long ago as 1959, John McCarthy pointed out that problem-solving strategies would really be far more understandable, more flexible, and easier to modify if they were first dissected out and made explicit. Unfortunately, this is easier said than done: it leads right back to the quandary of the general problem-solver.

On the other hand, there have been some interesting attempts in recent years. For example, in his work on TEIRESIAS, a knowledge acquisition subsystem for MYCIN, Stanford's Randall Davis included meta-rules that could adapt rule selection strategy to the nature of the problem. Meanwhile, Stanford's Michael Genesereth has devel-

Artificial Experts

It may seem paradoxical that mimicking a highly trained human expert is so much simpler than imitating an everyday ability such as language. But in fact, expertise is easier *because* it is specialized, because it focuses on narrow classes of problems.

An example is the first expert system, DENDRAL, developed at Stanford by Edward A. Feigenbaum and his colleagues in the late 1960's. DENDRAL simply advised chemists on the structure of unknown compounds, largely through the use of nuclear magnetic resonance and mass spectral data. But within that one domain it did rather well. For some families of compounds it could outperform the human specialists; a variant, GENOA, is now widely used in chemical laboratories.

A more sophisticated program, MYCIN, followed in the mid-1970's. MYCIN advised physicians on the diagnosis and treatment of blood and meningitis infections, again doing just about as well as the human experts. It also helped forge a lasting relationship between the medical community and the AI community: much of the work on expert systems has been done by a nationwide group of physicians and programmers working through Stanford's SUMEX-AIM network.

MYCIN was likewise influential in terms of its structure, which set the pattern for virtually all the expert systems that have followed. First, general information about the problem domain—in this case, infectious diseases—was encoded in a knowledge base of about 500 rules. (The rules themselves were in the form of IF-THEN statements, such as, "IF (i) the infection is meningitis and (ii) organisms were not seen in the stain of the culture and (iii) the type of infection may be bacterial and (iv) the patient has been seriously burned, THEN there is suggestive evidence that *Pseudomonas aeruginosa* might be one of the organisms causing the infection.")

A second MYCIN database contained information about the specific patient, including such things as age, sex, or test results. An "inference engine" then operated on the knowledge bases to draw conclusions or ask questions. Finally, and perhaps most importantly for MYCIN's credibility, there was a system for explaining the conclusions so that the users could evaluate MYCIN's advice.

In the wake of MYCIN have come systems such as INTERNIST-1 (internal medicine), PROSPECTOR (geological exploration), R1 (computer layout and configuration), PUFF (interpretation of pulmonary tests), and a host of others. Moreover the pace is picking up: a number of expert systems have gone commercial and venture capital is pouring into the field (*Science*, 24 February, p. 802).

This seems a little surprising at first, since even the enthusiasts have to concede that present-day expert systems are sharply limited. They are narrow: as Stanford's Bruce Buchanan says, "We operate, reluctantly, with a 'closed world' assumption—that nothing outside of the program is relevant." They are shallow: having no general principles, just thousands of rules of thumb, they are unable to infer missing knowledge. And perhaps most important, they do not learn: their "expertise" has to be put in by hand.

But then, human experts can be pretty narrow too, and as the above list indicates, there are a surprising number of well-defined domains where expert systems are appropriate.

Moreover, while the development of a new expert system can hardly be called easy, researchers have gotten the development time down to a reasonable 1 man-year. (Feigenbaum, a man solidly in the engineering camp of AI, talks about "knowledge engineers" who "mine" the knowledge from the expert's heads.) One important step in that direction was EMYCIN, which consists of the inference engine and explanation subsystem of MYCIN; the knowledge base is left empty to be filled in by rules from a new domain. More recently, Daniel G. Bobrow and his colleagues at Xerox Palo Alto Research Center have been trying to ease the critical shortage of knowledge engineers with LOOPS, an experimental knowledge programming system designed for easy learning by non-AI programmers.—**M.W.**

oped a meta-level control language as part of an effort to create programs capable of introspection.

Another set of issues cluster around the banner of *nonstandard logic*. For instance, how does one automate the reasoning process when the knowledge base includes statements like "He *believes* this," or "She *knows* that"? With symbolic logic it was easy, because the truth or falsity of a statement only depended upon its structure. But, here, truth or falsity depends upon meaning. Another example, strongly emphasized by Minsky in recent years, is that AI's focus on logical deduction ignores people's common sense ability to retract a conclusion upon further information. Technically referred to as the problem of "non-monotonic logic" it is almost invariably known as Minsky's "Dead Duck" challenge: If all ducks can fly and if Charlie is a duck, he points out, then Charlie can fly—unless Charlie is dead, in which case he cannot fly.

Finally, there is the matter of *reasoning with uncertainty*. A fact or a definition is absolute, but a rule-of-thumb is only plausible. So how much reliance can a heuristic program place in its own conclusions? In the case of MYCIN, weighting factors were assigned to the rules and were used to generate a numerical estimate of the conclusion's reliability. But it was not always clear what the final numbers meant. Efforts have been made to apply classic probability theory, without much success: most of the time the uncertainty in our knowledge is less a question of randomness than of vagueness. Several important steps have been taken to broaden probability theory, but still, no one really understands how a program is supposed to proceed in the face of ignorance—or how it can even learn to recognize the limits of its knowledge.

An even more fundamental question is whether reasoning ability is really all that important to intelligence. Consider the experiments of Herbert Simon and his colleagues on the way experts and novices solve physics problems (*Science*, 20 June 1980, p. 1335). Novices seem to go at it just as an AI program would. They work out elaborate strategies, complete with goals and subgoals, and solve every equation in sight. At each step, they ask, "What do I do next?"

Experts, on the other hand, seem to just look at a problem and see it whole: "AHA!" Whatever is going on, it is less a matter of reasoning than of recognition, of pattern matching on a huge store of memory and experience. Appropriate problem-solving sequences are simply there when they are needed. Certainly experts do use conventional reasoning, but only to solve unfamiliar problems. Once again it seems that knowledge, not reasoning, is power.

Acquisition: Until fairly recently, learning was not a major issue in AI. People were much more concerned with getting performance out of the knowledge bases they had. But several things have now brought learning to the fore.

On a purely practical level, the people building expert systems have to live with the fact that little human expertise is really codified, and that the experts are not necessarily doing what they think they are doing. Getting their knowledge into a machine is a slow, painstaking process. It would be nice if the machine could help.

More generally there is AI's quest for a mastery of "common sense," which

Is reasoning ability really all that important to intelligence?

seems to consist of massive expertise about the world in general. Quite aside from questions about knowledge representation and reasoning ability, building a machine with common sense means building a knowledge base containing millions of rules and facts—which is impractical unless the system can learn for itself.

And finally, there are deep theoretical problems: What *is* learning? How can a machine modify its knowledge according to experience? How can it be taught to learn from its mistakes?

Not surprisingly, the modern work on machine learning has tended to focus on the acquisition of knowledge, as opposed to the acquisition of motor and cognitive skills. [This may be an unconscious reflection of the distinction between "fact" memory and "skills" memory that has recently been made by psychologists (*Science*, 23 December 1983, p. 1318); there is some suggestion that the latter is inherently nonsymbolic.] Moreover, the tendency is to emphasize knowledge as a prerequisite. The idea is that machines, like humans, will learn best when they learn slowly—when they relate each new concept to what is already known instead of trying to organize a whole mass of facts by some instantaneous gestalt. In practice this means that the programs have to incorporate learning heuristics, rules of thumb to focus the program's attention, propose experiments, and choose the concepts to acquire.

Current work has also emphasized the need for a broad range of strategies. Programs that learn by example were first written back in the 1950's. Now they have been joined by programs that learn from instruction, that learn by analogy, and that learn by discovering concepts and categories on their own. A particularly intriguing example of the latter is AM, written by Stanford's Douglas B. Lenat. It works by a kind of natural selection process: starting from a knowledge base of concepts in set theory, it combines old concepts into new ones, and keeps those new concepts that seem to lead to a significant number of new results; it takes about 2 minutes of this to go from naïve set theory to arithmetic and the prime number theorem. (Unfortunately, it only takes an hour or so for AM to start studying such things as numbers that are simultaneously odd and even. A successor program, EURISKO, shows a bit more common sense.)

Perhaps the strongest single impression one gets from a survey of knowledge programming is that issues like representation, control, and acquisition are inextricably tied up in a knot—so interwoven that they might as well be the same thing. Is it a coincidence that the same could be said of human thought? Analogy, for example: people learn by analogy, they reason by analogy, they recall things by analogy, they even come up with creative new ideas by analogy.

In an invited talk on computer learning at the Washington, D.C., meeting of the American Association for Artificial Intelligence in August 1983, Lenat stressed this idea of underlying sources of power. "AI researchers have tended to focus on mechanisms and task domains instead of on why their programs worked," he said. There seems to be something very profound here. And perhaps it is time for a refocusing.

Perhaps so. The human mind is fluid, mercurial, nonlinear, continuously connecting and reconnecting, able to reorder itself and say "AHA!"—things that the careful representations and reasoning schemes devised by the AI researchers can do only with difficulty, if at all. One can reasonably ask whether a fundamental theory of intelligence is even possible within the symbolic processing model.

But then, there is only one way to find out. The AI researchers have their vision, and they are determined to push their ideas as far as they can.

—M. MITCHELL WALDROP