Soar: A Unified Theory of Cognition?

Originally a research project in artificial intelligence, the program seems to provide a general model of human thought

This is one of an occasional series of articles on cognitive science and artificial intelligence: the study of the mind as an information processor. A previous article (1 July 1988, p. 27) traced the history of Soar and how it works.

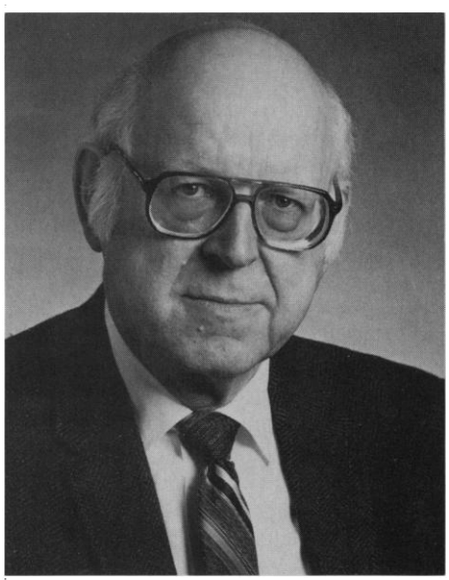
THE COMPUTER PROGRAM known as Soar made its public debut as a "unified theory of cognition" in the spring of 1987, when artificial intelligence (AI) pioneer Allen Newell was invited to give the annual William James lectures at Harvard University.

The Carnegie-Mellon University professor had been participating in the creation and development of Soar for nearly 5 years at that point, and he had become convinced that it was no longer just another AI project. In Newell's view Soar was the prototype for a new way of working in AI and cognitive psychology alike—an example of how researchers could go beyond "microtheories" that explain only one or two experimental results at a time, and instead devise theories of human cognition that encompass reasoning, learning, perception, motor control, language, cognitive development, emotion, and perhaps even such ineffable qualities as awareness, all within a single coherent framework.

In short, Newell was convinced that Soar could be a catalyst for a reformation in the way research is done in all the cognitive sciences. And the eight William James lectures accordingly became his manifesto: "Unified Theories of Cognition," he declared, "are within reach and we should strive to attain them."

What he was *not* asserting, he hastened to add, was "that there is somehow one such theory and we should all get together on it." Soar is neither perfect as it stands nor is it the only kind of unified theory one could imagine. Indeed, Soar has some notable predecessors as a unified theory, with perhaps the most influential being the Act* program developed by Carnegie-Mellon psychologist John R. Anderson in the 1970s and early 1980s to model memory and learning. So if anyone thinks he or she has a better idea, Newell said, then by all means develop it; the competition will benefit everyone. But in the meantime, Soar would serve as his exemplar, his own effort to show what a unified theory of cognition ought to be like.

Newell certainly seems to have struck a chord. The lectures themselves were reportedly greeted with at least one standing ovation. (They will soon be published as a book.) Moreover, the AI researchers and cognitive psychologists contacted by *Science* are generally agreed that Newell is right—that unification in some form or another is the way to go, and that Soar is indeed one of the most impressive attempts to date in how to achieve it. "It should make people sit up and take notice," says Anderson, who is already using Newell's lectures in his own psychology courses.



Allen Newell. *"Don't bite my finger—look where I'm pointing!"*

So what is Soar?

In his lectures, Newell starts his answer by first asking some fundamental questions about theories of cognition in general. Why, for example, should such a theory be written as a computer program at all? Why not use differential equations, say, or even plain English?

The reason, he says, is that a computer program is the most natural way to do what the theory itself has to do—namely, account for the information processing we do in our brains. Indeed, that is what the "cognitive revolution" of the past 30 years has been all about. AI, cognitive psychology, large segments of neuroscience, linguistics, philosophy, and even anthropology—all are predicated on the idea that thought can be understood in terms of the ebb and flow of information. And in that sense, computers and brains are fundamentally alike, however different they may be in organization and structure.

But then, exactly what kind of program should this hypothetical unified theory be?

One thing it clearly cannot be is a conventional programming algorithm, says Newell—at least not if "algorithm" means an unambiguous, precisely defined procedure that tells the computer what to do at every step along the way. What one wants instead is something a little more subtle. Think of a computerized spreadsheet such as *Lotus 1-2-3*: as it comes from the box it has structure, but no content. It simply provides a general template for numerical computation. Yet it is endlessly flexible in the sense that it can be filled with specific data and used for a near-infinity of projects. In much the same way, he says, a unified theory of cognition ought to be a cognitive *architecture*. It ought to be a generalized framework that makes learning, reasoning, and all the rest possible. And yet it ought to be endlessly flexible in the sense that it can be filled with specific knowledge that allows it to function in a near-infinity of situations in the world.

Soar, of course, is precisely such an architecture. Developed in the early 1980s by Newell and his former students John E. Laird, now at the University of Michigan, and Paul S. Rosenbloom, now at the University of Southern California, it was originally an effort to address some long-standing problems in the way AI programs are engineered. Indeed, one of the first things Newell, Laird, and Rosenbloom did with Soar was to test it on a wide variety of classic AI tasks, ranging from games and puzzles to such knowledge-intensive "expert system" tasks as medical diagnosis and computer hardware configuration.

As Soar successfully mastered those tasks, however, and as Newell, Laird, Rosenbloom, and their students began to move on to more difficult tasks such as language understanding and various kinds of learning, they became increasingly convinced that Soar had exactly what was needed for a general theory of human cognition. For example:

■ **Cognition as problem-solving.** Soar is basically a program that solves problems—any kind of problem. In formal terms it will

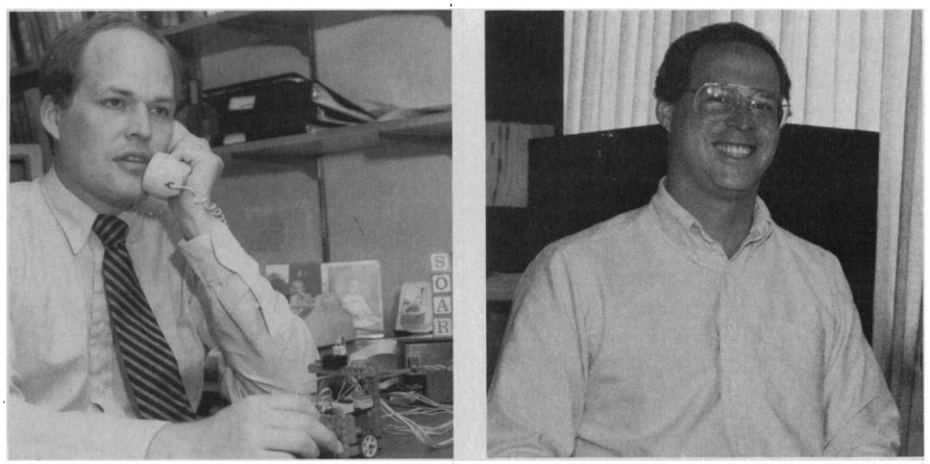
start from a given initial state (say, a certain configuration of pieces on a chess board), and then systematically search for a sequence of actions that will take it to a given final state (the opponent's King in checkmate).

More generally, says Newell, Soar fulfills a basic requirement of any intelligent system: its reasoning is directed in pursuit of its goals. There is ample evidence this kind of step-by-step reasoning is what happens in human problem solving: the model was actually first articulated back in the 1950s and 1960s by Newell, his Carnegie-Mellon colleague Herbert A. Simon, and computer scientist Clifford Shaw on the basis of extensive experiments on humans. Furthermore, says Newell, this kind of reasoning arguably applies even when the problem-solving is far from obvious—say, when we contemplate a sunset. Even then, we go from an initial state (sensory input from our eyes) to a final goal state (a comprehension of what we are seeing—sky, trees, clouds, washes of color, a ruddy sun, peace, beauty . . .). That journey from input to comprehension may be very fast. But it does not happen spontaneously. Indeed, in other situations it can be very slow and confusing, especially when we encounter something we have never seen before. And that task of finding our way is precisely what is meant by problem solving.

■ **Working memory.** If we think of a problem-solver as moving from mental state to mental state in search of a goal, then he, she, or it obviously needs some way of keeping track of what the current state is. Humans keep this information in a storage area known as working memory, which roughly corresponds to the set of things we are paying attention to at any given moment. (An older name is "short-term memory.") Working memory seems to function rather like a mathematician's blackboard, in that it holds not just current information, but partial results of problem-solving that might be useful later. Soar, like many other AI programs, uses a very similar storage area for this purpose. And by no coincidence, this area is also referred to as working memory.

■ **Long-term memory.** In addition to keeping track of the current situation, a problem-solver also has to make choices about what to do next. And that means in turn that it has to have access to facts, experiences, skills, and know-how—knowledge that it can use to make those choices intelligently.

In humans this sort of information resides in a storage area known as "long-term memory." In Soar it exists as a mass of condition-action rules: "IF *this* is the situation, THEN do *that*." On any given experimental run the program typically contains hundreds of such rules, each of which is constantly scanning the contents of working memory in search of items that correspond to the conditions on its IF side. Whenever one of the rules does detect a match, it "fires" and issues the commands on its THEN side. The program obeys, and thus moves step by step toward a solution.



Soar's co-creators. *The Soar research effort, which now involves some 35 people nationwide, is led by Newell, John Laird (left), and Paul Rosenbloom (right).*

The parallels between human memory and Soar's rule-based memory are not instantly obvious, but are suggestive nonetheless. Originally introduced by Newell and Simon in the late 1960s, rule-based programming was explicitly intended as a way to model human problem-solving behavior. In the two decades since then, moreover, it has been widely used by cognitive psychologists and AI researchers alike, not least because the rules provide such a convenient and flexible way of representing knowledge. A fact, for example, might be written as "IF object X is a crow, THEN X is probably black." A bit of procedural know-how might become "IF the goal is to start the car, THEN begin by putting the key in the ignition."

What Newell does in making the leap to a unified theory of cognition is simply to take the analogy literally. He envisions each rule as modeling the stimulus-response behavior of a network of neurons. Indeed, he prefers to think of it not as a rule per se, but as an individual piece of memory. To fire a rule—"IF X is a milk-chocolate Easter bunny, THEN X will taste creamy and sweet," for example—is to access that memory.

Now, in ordinary rule-based programs, such as expert systems, this rules-as-memory interpretation would not be viable, says Newell. The problem is that only one rule can be in control at any given time. Indeed, AI programmers have had to devise all kinds of schemes for resolving conflicts when several rules want to take control at once. In Soar, however, a modification first implemented by John Laird for his 1983 thesis allows *all* the rules to have their say at once; only afterward does the program weigh all the suggestions and make a decision. In effect, says Newell, the rules fire in parallel—which is exactly what one needs to do to emulate the massively parallel architecture of the brain, where billions of neurons are always operating simultaneously.

Furthermore, says Newell, the rules-as-memory interpretation is not just qualitative. If a rule really does correspond to the action of a network of neurons, then it ought to take about 20 milliseconds to fire—that is, about ten times longer than the typical response time of an individual neuron. And given that, he says, the Soar model of problem-solving makes testable predictions for the time scale of all kinds of higher level cognitive activity.

As an example, imagine a simple stimulus-response task such as pressing a button when a light goes on. Using the Soar model together with the 20-millisecond time scale for the firing of individual rules, Newell calculates that the response time should be about 220 milliseconds. In actual experiments the times average about 200 milliseconds. All in all, says Newell, he and his student Bonnie John have modeled 27 such stimulus-response tasks from the experimental literature, with an average deviation between theory and experiment of only 12%.

In short, concludes Newell, "Soar is to be used not only as a theory of problem-solving, but as a detailed model of microcognition."

■ **Autonomy and adaptability.** Yet another fundamental requirement of an intelligent system is that it should not depend upon a programmer to tell it what to do. The world is too complicated and unpredictable, and there is no way that a programmer can anticipate every situation in advance. So the system has to be able to decide for itself,

based on the needs of the moment.

According to the common perception of computers, of course, this is impossible. Computers are held to be blind idiot savants, perfectly capable of dunning an innocent customer with bill after bill for $0.00. Right or wrong, they do exactly what their programmers tell them to do and nothing more. Even in AI, where a great deal of effort has gone into making programs more flexible than that, a typical rule-based system will abruptly slam to a halt when it encounters a situation that its rules do not cover. It has no way of knowing what to do next.

In Soar, however, a second innovation implemented by Laird builds in flexibility and resourcefulness from the start. Whenever Soar comes up against an impasse where its rules do not tell it what to do next, it automatically sets itself a subgoal—"solve this impasse"—and then brings all its problem-solving abilities to bear. Its behavior is reminiscent of the way human doctors would react to a patient with unfamiliar symptoms: instead of just throwing up their hands because they could not make a diagnosis instantly, they would try to work around the roadblock by reasoning about basic anatomy and physiology, or by recognizing an analogy to more familiar cases, or even by experimenting with therapies on a trial-and-error basis. They would try *something*. And in much the same way, Soar will draw on any body of knowledge that seems useful, any set of facts and procedural tricks that will help it achieve a resolution. Thus, says Newell, "Soar does not have to be programmed to behave."

■ **Learning from experience.** A final requirement on any intelligent system is that it be able to learn from experience. Otherwise, it might be condemned to repeat the same actions and the same mistakes over and over again like a stereotypical computer.

In Soar, however, this kind of learning occurs automatically. Anytime its subgoaling mechanism resolves an impasse, Soar simply remembers how. That is, it creates a new condition-action rule that tells it what to do the next time it encounters a similar situation, using a process known as "chunking." And once that rule is in place, Soar never has to deal with that impasse again.

This approach to learning immediately gives Soar some interesting properties, says Newell. The very fact that it learns only when it is solving a problem means that its learning is highly individualistic; like humans (and unlike a preprogrammed computer), it remembers only what it has experienced and only what it has focused on. Furthermore, and for the same reason, Soar possesses a human-like ability to be idiosyncratic and illogical; there is nothing that says that its experience will lead to rules that are mathematically precise and completely consistent with each other.

But even more interesting, says Newell, is that this one simple chunking mechanism seems to account for a wide range of human-like learning behaviors. A prime example is the improvement of mental and physical skills with practice, a phenomenon that was studied by Soar co-creator Paul Rosenbloom for his 1984 thesis research. In humans, skill acquisition is characterized by "the power law of practice," which says that a person's performance on a given task will almost invariably speed up as some power of the number of practice trials. (The power varies from task to task.) Indeed, this is one of the most solidly established regularities in all of cognitive psychology; it has been demonstrated in tasks ranging from the reading of upside-down text to the rolling of cigars by hand in a cigar factory.

What Rosenbloom was able to show, says Newell, is that this purely empirical finding has a natural explanation in terms of chunking. Chunking, by its very nature, takes a complex, slow piece of problem-solving and replaces it with a simple, fast stimulus-response reflex: "IF *this* is the situation, THEN do *that*." Moreover, these new rules accumulate every time Soar works through a practice session. The upshot is that the program's performance will get faster each time it repeats the task—at a rate that does indeed approximate a power law.

"Soar," says Newell, "does not have to be programmed to behave."

Another, more subtle form of learning has to do with recovery from error. Soar—like humans—is perfectly capable of drawing the wrong lessons from its experience. It sometimes creates rules that are too specific, or else so general that they are just plain wrong. Worse, Soar has no way to delete an incorrect rule from memory, any more than a human can forget his or her name just by deciding to. So the potential is catastrophic: one can all too easily imagine the program piling up errors until its reasoning resembles a kind of hopelessly muddled schizophrenia.

Not surprisingly, the problem of how to deal with incorrect knowledge is still a very active area of research in the Soar project. Recently, however, Newell and his student Rex Flynn have been exploring a way for Soar to make an end run around the problem. Assuming that Soar can recognize the error in the first place, they have shown that the program can reformulate its problem-solving strategies so that it never again gets into a situation where the incorrect rule can fire. The rule is still there, but it is impotent. Intriguingly, this approach seems to produce a style of learning similar to that of very young children: successive cycles of strategy reformulation lead Soar through stages of cognitive development much like those studied by the pioneering child psychologist Jean Piaget. "And this opens up the possibility of modeling the transition mechanisms in detail," says Newell.

So what can one conclude from all this? Obviously, that Soar is an impressive piece of research: "The thing I like is that [the Soar group] tries to solve a whole bunch of hard and interesting problems simultaneously," without a lot of ad hoc additions, says AI researcher John McDermott of the Digital Equipment Corporation. "It's a very responsible piece of science."

And yet, one also has to conclude that Soar is still a long way from being a satisfactory unified theory of cognition. As Newell himself is the first to admit, there are plenty of things about human cognition that Soar has *not* accounted for, with two obvious examples being emotion and consciousness. Nor does it have more than a rudimentary ability (so far) to cope with the external world via sensors and manipulators. And even in areas it supposedly does cover, such as learning, the accounts are often quite sketchy.

Nonetheless, Newell sees a great deal of virtue in sticking with Soar as it is and seeing where it leads. "This is a programmable system," he says. "And when you a reach a problem, you could always just patch in another module to fix it. But then, as the system became more and more baroque you would lose predictability because it now has arbitrary degrees of freedom." If one listens to the architecture instead, he says, the structure of the program may suggest solutions that were never anticipated by its creators. This has actually happened in several cases already, including the recovery from error problem mentioned above. "So you say, Wow! And you tend to believe the architecture."

In sum, then, it seems safe to say that Soar has a long way to go—and that it could go a long way. Whether one agrees with its assumptions or not, whether it ultimately succeeds or not, the creators of Soar have at least tried to lead the way by example. As Newell himself is fond of saying, in a quotation from the late neuroscientist Warren McCulloch: "Don't bite my finger—look where I'm pointing!"

■ **M. Mitchell Waldrop**