In the early 1980s, a trio of researchers proposed a new problem-solving architecture that drew on AI research dating back to Allen Newell and Herbert Simon's pioneering work in the 1950s. As defined by Newell, John Laird, and Paul Rosenbloom, the SOAR architecture is based on specific theories about the computational basis of general intelligence. Because many of these theories were devised to account for the empirical results of studies of human problem solving, SOAR is also a theory of the human cognitive architecture.
However, Soar also has much in common with other AI approaches. It views all problem solving as problem-space search, it supports a variety of weak methods (such as means-ends analysis and depth-first search), and it is based on a production system. But most important to the designers of knowledge-based systems, Soar views intelligent agents as having, accessing, and applying all relevant knowledge.

Knowledge-based systems suffer from three major problems: They are too brittle, it is difficult and time consuming to acquire knowledge for them, and their results are hard to share with other researchers and developers. Researchers have used Soar to address each of these problems.

Brittleness

Soar provides a nonprocedural way to specify system behavior—by encoding independent sources of knowledge that determine system behavior when combined dynamically at runtime. This allows more flexible behavior because the specific problem-solving steps are not predetermined when the system is designed.

Soar also supports metalevel reasoning of unlimited complexity, letting systems recursively reason about all aspects of problem solving. If a Soar diagnosis system cannot explain all symptoms with the available knowledge, it can formulate a secondary problem: discovering why the symptoms cannot be explained. It might then discover that its strategy was incorrect and that it failed to consider a pertinent disorder. The levels of self-reflection are theoretically infinite: Soar could reason about its technique for discovering why the symptoms could not be explained, reason about this reasoning, and so on.

Soar can also smoothly integrate multiple problem-solving methods and knowledge sources. It can integrate strong (expert) and weak methods so that a Soar system can use strong methods when appropriate and weak methods when a strong method is insufficient or unavailable. Soar can also integrate deep and surface (compiled) knowledge, so that deep knowledge is used only when compiled knowledge cannot resolve a problem. Soar systems can thus use the most relevant domain- or problem-specific strategies to solve most problems, falling back as necessary on weaker (and often less efficient) strategies or less compiled knowledge.

Knowledge acquisition

Soar assists knowledge acquisition in two ways. First, the abstract computational elements of Soar’s theory of intelligence lead to a stratified approach to system building: The system is first described at the knowledge level, then in terms of the problem spaces it uses to perform computation, and finally at the implementation level. The knowledge-level description specifies the knowledge that must be acquired, while the problem-space level provides a uniform way to organize this knowledge. This uniformity leads to a unique knowledge acquisition strategy that converts descriptions of domain knowledge directly into problem-space constructs.

Second, Soar’s theory of learning lets us acquire recognition (compiled or surface) knowledge directly from deep knowledge. By encoding deep knowledge and the strategies to explicitly use it, we avoid the need to acquire and code thousands of rules. Soar automatically builds the necessary recognition knowledge as it solves problem instances.

Sharing of results

Soar provides a uniform, implementation-independent language for describing system states (as a set of problem spaces) and the knowledge required to use them. Such abstract computational descriptions let researchers working outside Soar understand and use the results of Soar research.

The IEEE Expert series

Each article in this special series focuses on how Soar solves specific problems and addresses issues important to the design of intelligent systems. In this issue, Jack Smith and Todd Johnson introduce Soar and describe a stratified approach for designing knowledge-based systems. Gregg Yost then discusses knowledge acquisition issues, including tools designed to facilitate Soar programming. In the third article, David Steier examines the integration of multiple sources of knowledge in tasks such as design and natural-language comprehension.

In later issues, Todd Johnson et al. will discuss the integration of multiple problem-solving methods and the integration of weak and strong methods, planning, abstraction, and model-based reasoning, and Charlie Dulan and Paul Rosenbloom will compare Soar to other architectures and describe possible future versions.

Soar is actively supported public-domain software. During the past ten years, Soar research has spread to about 73 institutions worldwide, and yearly workshops in the US and Europe offer a chance to collaborate and exchange ideas. For more information about Soar, or to find out how to obtain the latest copy, send e-mail to soar-request@cs.cmu.edu

Suggested reading


Jack W. Smith is an associate professor of pathology and computer and information science and director of the Division of Medical Informatics at Ohio State University. His research interests include task-specific and cognitive architectures in knowledge-intensive domains, abductive problem solving, and medical decision support systems. He earned his BS in physics from Virginia Polytechnic Institute, his MD from West Virginia University Medical School, and his MS and PhD in computer and information science from Ohio State University. Smith is on the editorial board of Artificial Intelligence Medicine and is a member of the IEEE Computer Society, ACM, AAAI, the Cognitive Science Society, the Society for Medical Decision Making, and the American Medical Informatics Association.

David M. Steier is a research scientist at the Engineering Design Research Center and the School of Computer Science of Carnegie Mellon University. His research focuses on the development of useful AI techniques, especially integrated problem-solving and learning architectures, to problems in engineering design and human-computer interaction. He received his BSc from Purdue University and his MS and PhD from Carnegie Mellon University, all in computer science. He is a member of AAAI. He has also worked at the Engineering Design Research Ctr., Carnegie Mellon University, Pittsburgh, PA 15213.

Todd R. Johnson is an assistant professor of pathology in the Laboratory for Knowledge-Based Medical Systems at Ohio State University. His current research interests include flexible problem solving, modeling the acquisition of abductive problem-solving skill, and reasoning with external information. He earned his BS, MS, and PhD in computer and information science from Ohio State University. He is a member of AAAI.

Smith and Johnson can be reached at Ohio State University, Division of Medical Informatics, 571 Health Sciences Library, 376 W. 10th Ave., Room 571, Columbus, OH 43210; e-mail, johnson.25@osu.edu or smith.30@osu.edu