

## Guest Editors' Introduction

# Case-Based Reasoning

Janet Kolodner, Georgia Institute of Technology  
William Mark, Lockheed

**R**EASONING IN ARTIFICIAL INTELLIGENCE has traditionally involved writing and customizing rules or models to solve problems. But there is another paradigm that has proven effective in many experimental and applied systems: Case-based reasoning "remembers" previous problems and either adapts their solutions or uses their outcomes to evaluate new cases.

There are two kinds of case-based reasoners: problem solvers and interpretive reasoners. A problem solver adapts old solutions to new problems; old solutions can provide almost-correct solutions and can warn of potential mistakes or failures. An interpretive reasoner uses cases to evaluate or justify new situations, much as lawyers use previous cases as arguments for categorizing new situations. Interpretive reasoners can evaluate solutions when no clear-cut methods are available and can interpret situations whose boundaries are open-ended or fuzzy.

The first case based systems were autonomous problem solvers, but recent systems are interactive external memories for users who actually solve the problem. Interactivity shifts much of the burden of adapting old solutions from the reasoner to the user, but it also forces developers to consider how people interact with computers. Thus, *IEEE*

*CASE-BASED REASONERS "REMEMBER" PREVIOUS PROBLEMS AND USE THEIR KNOWLEDGE OF WHAT WORKED BEFORE TO SOLVE OR EVALUATE NEW PROBLEMS. THIS SPECIAL SERIES WILL EXAMINE THE BUILDING AND DEPLOYMENT OF REAL SYSTEMS, PLUS ISSUES OF HUMAN-COMPUTER INTERACTION AND THE SYSTEM LIFE CYCLE.*

*Expert's* special series on case-based reasoning examines not only the building and deployment of real systems but also issues of human-computer interaction and the system life cycle. Among the key questions are: How do we choose a good case? How can we merge several cases to form a new solution? How does a reasoner know which parts of a previous case to focus on? How can we integrate a case-based reasoner with a rule-based system?

### The case for case-based reasoning

In traditional reasoning systems, building and using reasonably complete domain models or acquiring large bodies of rules can be extremely difficult due to inexpressive

representation languages, incomplete or uncertain domain understanding, or simply the sheer volume of knowledge. On the other hand, by using its memory of what worked in the past (and what didn't), a case-based reasoner can

- focus on a problem's important features, generally those that led to failure or success in a previous case;
- make assumptions and predictions to solve problems in domains it doesn't fully understand;
- estimate whether a solution that failed in a previous, similar situation will fail now;
- reuse old reasoning about how a failure was repaired or could have been bypassed to avoid repeating previous mistakes;
- ease knowledge acquisition, since cap-

turing actual experiences of past successes and failures is fairly easy;

- evaluate solutions when no algorithmic method is available, which is particularly helpful when the existence of many unknowns would make traditional evaluation onerous; and
- interpret open-ended and ill-defined concepts, especially those that are themselves composed of cases rather than other concepts (such as in law or biological classification).

Case-based reasoning is also a learning methodology. Learning in AI usually means learning generalizations through either induction or explanation. A case-based reasoner can certainly induce generalizations based on its ability to detect similarities between cases, but this is only part of how it learns. Case-based reasoners learn mostly by accumulating and indexing cases. New cases give a reasoner more familiar contexts in which to solve problems or evaluate situations: Clearly a reasoner whose cases cover more of a domain is better than one that covers less, and one whose cases cover successes and failures is more helpful than one that covers only successes. New indices help a reasoner fine-tune its recall apparatus so that it remembers cases more appropriately. Thus, a case-based reasoner becomes more efficient and competent by increasing its memory of old solutions and adapting them, deriving better answers more easily than it could if it had to derive new answers each time. Although case-based reasoners still must evaluate their proposed solutions, not having to repeat time-consuming computations and inferences is a considerable advantage over traditional systems.

Of course, a case-based reasoner that relies on previous experiences without sufficient validation can retrieve inappropriate cases, wasting precious problem-solving time and even making costly errors that might have been avoided with more incremental methods. Also, in hybrid reasoners, cases might bias the reasoning too much, suppressing compositional details or letting the reasoner ignore important novel problem features.


### **The special series**

This issue features three articles on case-based reasoning; more will appear in future

issues. The first article describes Cascade, an interactive system that helps technical-support engineers (a company's "help desk") solve the problems of customers whose hardware or software has stopped working. Cascade has a case library of failures of VMS device drivers, and it suggests solutions to new failures. Cascade's retrieval algorithm — validated retrieval — is particularly important.

The second article examines an early attempt to build a case-based advisory system for a complex domain. Archie is intended to help architects understand and solve conceptual design problems. The

One of the more interesting lessons from Clavier and Cascade is social. In a normal working environment, both systems make every user's experiences available to every other user. The case library is a sort of corporate memory, recording the new things that each user encounters. This may be one of case-based reasoning's most useful functions: While the system helps people do their jobs more accurately and efficiently, it gives the entire company the accumulated knowledge of its employees, making employee turnover less catastrophic and making it possible to analyze the effects of new procedures before they are put into effect.



## **CASE-BASED REASONERS BECOME MORE EFFICIENT AND COMPETENT BY INCREASING THEIR MEMORY OF OLD SOLUTIONS AND ADAPTING THEM, DERIVING BETTER ANSWERS MORE EASILY THAN IF THEY DERIVED NEW ANSWERS EACH TIME.**

designer describes a problem to Archie, which retrieves and presents cases to the designer. Each case teaches one or more lessons relevant to the issues at hand. Since Archie is a more experimental system, many interesting lessons were learned from its behavior and the reactions of its users.

The third article describes Clavier, which Lockheed is using to determine the placement of parts made of composite materials in an autoclave (a large convection heater). The heating rate of all parts put in an autoclave must be controlled carefully, but their number, shape, and placement can cause significant nonlocal variations. Before Clavier was deployed, operators loaded the autoclave manually based on records of previous successful and unsuccessful combinations. Now, Clavier provides interactive support, using cases to propose load configurations and multiloading plans. One of its key advantages is that it learns, becoming more competent as it acquires new cases.



**Janet Kolodner** is a professor in the College of Computing at the Georgia Institute of Technology. She received her PhD in computer science from Yale University in 1980. Having pioneered the development of case-based reasoning, she is concentrating on applying that method to design, and on the implications of CBR for decision aiding, education, and creative problem solving. She is editor-in-chief of *The Journal of the Learning Sciences*.



**William Mark** is chief scientist of information and computing science at Lockheed Palo Alto Research Laboratories and a member of the executive editorial board of *IEEE Expert*. His research interests include artificial intelligence, software synthesis and reuse, and

massively parallel hardware and software. Mark received his BS and MS in electrical engineering and computer science in 1973 and his PhD in computer science in 1976, all from the Massachusetts Institute of Technology. He is a member of AAAI, ACM, the Association for Computational Linguistics, and the American Institute of Aeronautics and Astronautics. Readers can reach him at Lockheed Palo Alto Research Laboratories, O/96-20, B/254E, 3251 Hanover St., Palo Alto, CA 94034-1191; Internet, mark@sumex.stanford.edu