Explanations in Knowledge Systems

The Role of Explicit Representation of Design Knowledge

B. Chandrasekaran, Ohio State University
William Swartout, University of Southern California

THE FOLLOWING TWO ARTICLES examine explanation in expert systems. The unifying idea in these projects is of general importance to explanation: The more explicitly we represent the knowledge underlying a system's design, the better its explanations.

EXPLANATIONS OF A KNOWLEDGE SYSTEM'S CONCLUSIONS CAN BE AS IMPORTANT AS THE CONCLUSIONS THEMSELVES. THE UNIFYING IDEA IN THE NEXT TWO ARTICLES IS OF GENERAL IMPORTANCE TO EXPLANATION: THE MORE EXPLICITLY WE REPRESENT THE KNOWLEDGE UNDERLYING A SYSTEM'S DESIGN, THE BETTER ITS EXPLANATIONS.

Knowledge and methods of using knowledge are the fundamental elements of knowledge systems. Much knowledge-system research has been concerned with developing increasingly explicit representations of these elements to support increasingly sophisticated techniques for knowledge acquisition, system building, and explanation. From an explanation standpoint, explicit representations of knowledge and method enable a knowledge system to examine its own structure and produce explanations from the same structures used for reasoning.

The idea that explicit representations facilitate explanation was recognized early on (for example, in Mycin[1]). It soon became evident that higher level strategies played a role in a knowledge system's ability to solve problems. Researchers began to explicitly represent problem-solving strategies (methods or plans) for using knowledge to solve problems. Examples here include Digitalis Advisor,[2] MDX,[3] and Neomycin.[4]

Another important idea for knowledge systems is that we can obtain problem-solving knowledge from other, more general knowledge. In justifying its conclusions, a knowledge system might need to justify the knowledge used to reach them. This in turn requires access to the more general knowledge that produced the system's knowledge. (Although such general knowledge is sometimes called "deep" knowledge, or "first principles," it is not better knowledge; it is simply more general, that is, useful for more purposes rather than for a specific problem type.) Knowledge systems based on explicit representations of knowledge and method, with information about how and from what their knowledge was obtained, are the foundation for producing good explanations.

Each type of explicit knowledge makes specific kinds of explanation possible. Also, each such representation makes explicit an aspect of the design of the knowledge system itself. For example, representing generic problem-solving methods or strategies is a way to make explicit the strategies that are often implicit in expert-system knowledge bases. Similarly, representing the more general knowledge used to derive specific pieces of knowledge in the knowledge base involves making another aspect of the design explicit, namely, where the knowledge in the knowledge base came from. Thus, the operational principle at work here calls for increasingly explicit design knowledge.

0885/9000/91/0600-0047 $1.00 © 1991 IEEE

Tasks and knowledge

Let's say system S performs a task T using knowledge K. (We would normally use S_T and K_T to indicate that S is a system that solves T, and that K is the knowledge related to T, but we dispense with the additional notational complexity here.) For example, Mycin is a problem-solving system (S) that performs the task of consulting about infectious diseases (T) using the rules in its knowledge base (K). A task is a collection of problem instances of a certain type. For example, Mycin can solve a number of instances of consultation problems in infectious diseases.

$\Delta(S)$ is the knowledge needed to design S, and $R(S)$ is the design record (the record of how $\Delta(S)$ was used to design S). $\Delta(S)$ typically includes substantial knowledge about the domain's subject matter, the nature of the task, the range of appropriate strategies, the architecture of S, how the parts of S accomplish T, and the origin and range of applicability of K. $R(S)$ would consist of the various design documents recording the design process. In the Mycin example, Δ(Mycin) is the designer's knowledge about the domain, the task, and AI architectures (most of which is never made explicit), while R(Mycin) explicitly captures a small part of its design using an informal notation (such as English or diagrams). Abstractly, knowledge-system design is a process that produces S from $\Delta(S)$, and $R(S)$ as a partial record of this process.

By the nature of design knowledge, $\Delta(S)$ can support the design of a range of systems of which S is a specific instance. For example, $\Delta(S)$ might contain a parametric family of strategies, one of which is instantiated in S. In the Mycin example, the same underlying knowledge of the domain, task, and strategies could be used to design variants of Mycin that perform different versions of the task or the same task in different ways.

Much expert-system research has emphasized the advantages of explicitly representing K. In fact, expert systems as a field is identified by its explicit representation of K and its application of various forms of inference to K to solve problems. While we need not explicitly represent $\Delta(S)$ to solve T, there are several advantages in explicitly representing relevant components of $\Delta(S)$, including how knowledge in them was used:

- Knowledge in $\Delta(S)$ and $R(S)$ about strategies and about K lets us explain the behavior of S and justify its conclusions.
- By operating on the representation of strategies in $\Delta(S)$, we can generate a range of behaviors of S.
- The explicit representation in $\Delta(S)$ of knowledge and strategies used by S makes system maintenance easier.

Of course $\Delta(S)$ is open-ended, so we cannot explicitly and formally represent all of it. However, as research in knowledge types, strategies, and architectures progresses, we will have an increasingly rich vocabulary to represent more and more parts of $\Delta(S)$.

EXPLICIT REPRESENTATIONS OF KNOWLEDGE AND METHOD ENABLE A KNOWLEDGE SYSTEM TO EXAMINE ITS OWN STRUCTURE AND PRODUCE EXPLANATIONS FROM THE SAME STRUCTURES USED FOR REASONING.

The accompanying articles

The following articles present results from two groups, both concerned with using knowledge about a system's design to explain it. In the first article, Michael C. Tanner and Anne M. Keuneke report on work at Ohio State University, which has concentrated on

- identifying appropriate strategy abstractions (the generic-task work);
- exploring the relation between task requirements and strategies available in S; and
- understanding the relationship between diagnostic-task knowledge and structure-function models of the device being diagnosed.

The work reported in the article focuses on producing strategic and task explanations and on justifying knowledge.

The second article comes from the Explainable Expert Systems project,[6-8] in which William Swartout and his associates have focused on

- distinguishing and providing explicit representations for several kinds of knowledge in $\Delta(S)$, including a domain model and general strategies;
- using an automatic program writer to create an explicit design record $R(S)$ that records how knowledge in $\Delta(S)$ was applied to create S; and
- capturing the "design" of explanations, that is, what the system was trying to say and how it was trying to say it.

In the terms used above, the EES project is concerned with two main tasks: the task T solved by the knowledge system S, and the task of constructing explanations $E(T)$ of S's performance. Let $\Delta(E(T))$ denote the design knowledge that supports the construction of the explanation $E(T)$, and let $R(E(T))$ be the record of how that design knowledge was used to construct a particular explanation. $R(E(T))$ captures what the explanation component tried to convey in a particular explanation, what explanation strategies it used to convey it, and what alternative strategies exist to get the same points across. This is exactly the information needed by the dialogue component to understand follow-on questions in context and to correct misunderstandings. In their article, Swartout, Cécile Paris, and Johanna Moore discuss knowledge justifications and issues related to presenting explanations and dialogue with users.

Acknowledgments

This work has been supported by the Air Force Office of Scientific Research (grant 89-0250), by the Defense Advanced Research Projects Agency (grant MDA 903-81-C-0335 and contracts F30602-85-C-0010 and F49620-89-C-0110), and by a NASA Ames cooperative agreement (number NCC 2-520). We thank Saul Amarel, Cécile Paris, and Michael Tanner for their comments on earlier drafts.

References

1. E.H. Shortliffe, *Computer-Based Medical Consultations: Mycin*, American Elsevier, New York, 1976.
2. W.R. Swartout, "A Digitalis Therapy Advisor with Explanations," *Proc. Fifth Int'l Joint Conf. Artificial Intelligence*

(*IJCAI 77*), Morgan Kaufmann, San Mateo, Calif., pp. 819-825.

3. B. Chandrasekaran and S. Mittal, "Conceptual Representation of Medical Knowledge for Diagnosis by Computer: MDX and Related Systems," in *Advances in Computers*, Vol. 22, M.C. Yovits, ed., Academic Press, New York, 1983, pp. 217-293.
4. W.J. Clancey and R. Letsinger, "Neomycin: Reconfiguring a Rule-Based Expert System for Application to Teaching," in *Readings in Medical Artificial Intelligence*, W.J. Clancey and E.H. Shortliffe, eds., Addison-Wesley, Reading, Mass., 1984, pp. 361-381.
5. B. Chandrasekaran, "Generic Tasks in Knowledge-Based Reasoning: High-Level Building Blocks for Expert System Design," *IEEE Expert*, Vol. 1, No. 3, Fall 1986, pp. 23-30.
6. W.R. Swartout and S.W. Smoliar, "On Making Expert Systems More Like Experts," *Expert Systems*, Vol. 4, No. 3, 1987, pp. 196-207.
7. J.D. Moore and W.R. Swartout, "Pointing: A Way Toward Explanation Dialogue," *Proc. Eighth Nat'l Conf. Artificial Intelligence* (*AAAI 90*), MIT Press, Cambridge, Mass., pp. 457-464.
8. W.R. Swartout, "Knowledge Needed for Expert System Explanation," *Future Computing Systems*, Vol. 1, No.2, 1986, pp. 99-114.

B. Chandrasekaran directs the AI group and is professor of computer and information science at Ohio State University. He is also editor-in-chief of *IEEE Expert*. His research interests address knowledge-based reasoning. Chandrasekaran received his BE with honors from Madras University in 1963 and his PhD from the University of Pennsylvania in 1967. Readers can reach him at 217 Bolz Hall, Ohio State University, 2036 Neil Ave., Columbus, Ohio, 43210; e-mail, chandra@cis.ohio-state.edu

William Swartout is director of the Intelligent Systems Division and an associate research professor of computer science at the USC's Information Sciences Institute. His research interests include explanation and text separation. He started and led USC/ISI's Explainable Expert Systems project, and he was the principal designer of the Xplain system at MIT. Swartout received his bachelor's degree from Stanford University, and his MS and PhD in computer science from MIT. Readers can reach him at USC/ISI, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292-6695.