A Practical Guide to Designing Expert Systems. Sholom M. Weiss and Casimir A Kulikowski. Rowman & Allanheld Publishers, 1984. 186 pp., $24.95.

Reluctantly, I must admit that this is a good book. Weiss and Kulikowski have admirably delivered what they promise: a simple, proven-effective means for building prototype expert systems. The authors have considerable experience and speak with authority. Their points concerning diverse problems, such as selecting applications, knowledge acquisition, and strategic issues such as controlling questioning are clear and useful. What I most like about this book is that it is not pretentious. It deals only with what the authors understand best about expert systems, and all of that is presented simply, with good examples. The book steers clear of academic arguments about knowledge representation, and this simplification seems appropriate for a practical engineer's guide.

As a basic guide for designing expert systems, the book offers the classification model as a common theme for describing how certain expert programs solve problems. A classification expert system is one that selects an output from a pre-enumerated list of possible solutions that is built into the program. Weiss and Kulikowski present this model in a simple way, describing CASNET, PROSPECTOR, DART/DASD, and similar systems as examples. Problem definition, elements of knowledge, and uncertain reasoning are treated concisely. The brief discussion of traditional problem solving methods, such as decision theory, is valuable. EXPERT, a production rule language, is illustrated by a hypothetical car diagnosis problem as well as a model for serum protein interpretation. Of particular interest is a description of the ELAS system for oil well log analysis, which integrates EXPERT with traditional analysis programs. The book concludes with an interesting, down-to-earth essay on the state of the art and consideration of the future.

But for all its good sense and clear exposition, the book has two important limitations. First, the classification model presented here is weakly developed; it applies only to the simplest problems. Much more is known about classification from studies of human problem solving. The authors ignore cognitive science studies altogether and so leave out basic ideas that are relevant to designing expert systems. Even more serious, the authors advocate a rule-based programming style that I am afraid may become the FORTRAN of knowledge engineering. So much knowledge is left implicit or is redundantly coded that modifications and extensions to the program will be expensive—just like maintaining FORTRAN programs. If we want to make knowledge engineering an efficient, well-structured enterprise, we can only hope that approaches like those used in EMYCIN, EXPERT, and OPS5 will soon die out. Examples from this book make my point.

I will consider the classification model first. It is noteworthy that the two AI researchers who first described expert systems in terms of classification— Kulikowski (Kulikowski, 1980) and Chandrasekaran (Chandrasekaran, 1984)—both had experience with pattern recognition research in Electrical Engineering. Some of the most informative parts of *Designing Expert Systems* relate expert system research to pattern recognition and decision analysis. What is lacking in this analysis is similar attention to the other fork of the evolutionary tree, studies of human problem solving in cognitive science. After all, the patterns of an expert system are not linear discrimination functions, they are concepts. Research concerning the nature of memory and learning of categories is relevant for designing expert systems. In particular, the hierarchical structure of knowledge, the nature of schemas as stereotypes, and the hypothesis formation process all have a bearing in how we design an expert system.

Certainly, in the language of EXPERT, Weiss and Kulikowski have taken a big step beyond EMYCIN by structuring knowledge in terms of findings, hypotheses, and different kinds of rules relating them. They list three kinds of rules: finding $\longrightarrow$ finding, finding $\longrightarrow$ hypothesis, and hypothesis $\longrightarrow$ hypothesis. Thus, the classification nature of the problem solving method is revealed as a mapping of findings onto hypotheses. Moreover, Weiss and Kulikowski describe search of this knowledge network independently, so inference knowledge is not mixed with process knowledge. But their analysis stops here. Weiss and Kulikowski are right to put forth the classification model as a scheme for structuring expert knowledge, but they have not made any attempt to relate it to what is known about experiential human knowledge.

Further analysis shows that there are common relations that underlie the rules (Clancey, 1984). For example, findings are related to each other by definition, qualitative abstraction, and generalization. Knowing this provides a basis for acquiring, documenting, and explaining finding/finding rules. Besides asking the expert, "Do you have any way to conclude about F from other findings?" the knowledge engineer could also say, "Do you know subtypes of F?" or "Given this numeric finding, do you speak in terms of qualitative ranges?" Similarly, hypotheses are related by subtype or cause. Rather than considering car failure diagnoses (an example developed in the book) as a simple linear list, the knowledge engineer can start with the assumption that the expert organizes his knowledge as a hierarchy of diagnoses.

The classification model can be further refined in several ways. First, a distinction can be made between heuristic classification and simple classification by direct matching of features (as in botany and zoology). The pre-specified solutions in expert systems are often stereotypic descriptions, not patterns of necessary and sufficient features. This has important implications for knowledge acquisition and ensuring robustness in dealing with noisy data. Second, emphasizing rule implication alone, Weiss and Kulikowski fail to mention

that findings are abstracted into problem categories (they call them only "intermediate hypotheses") or that hypotheses are refined into subtypes (they say that hypotheses can be organized in a taxonomy, but give no examples). Most importantly, they miss the idea that expert systems often solve a sequence of problems by classification. Common examples are: making a diagnosis and then selecting a repair, characterizing a patient stereotypically and matching this to diseases, and modeling a user's needs and satisfying them (see (Clancey, 1984) for further discussion). This more refined conception of classification problem solving—incorporating stereotypic solutions, abstraction/refinement, and decomposition of problems into a sequence of classifications—is of considerable practical value for designing expert systems.

Beyond this, Weiss and Kulikowski perpetuate the confusion that classification is a property of problems, rather than a problem solving method. Diagnosis is not inherently a "classification problem." Rather if the problem solver can abstract problem findings to select a previously known diagnostic solution, he can solve the problem by classification. Any kind of problem can be solved this way—planning, configuration, diagnosis—it just depends on building in solutions and presenting problems that fit known solutions.

By identifying classification with diagnosis and interpretation, Weiss and Kulikowski fail to consider perhaps the most appropriate use of the classification model—"catalog selection." This is the problem faced by a librarian, wine advisor, dietician, career planner, travel agent, *etc.* In catalog selection problems the solutions are generally well-specified, enumerable alternatives which fall into categories that are easily related to abstractions of the problem, for example, "this is a wealthy client (finding category) who wants to take a cruise (solution category)." Again, Weiss and Kulikowski's version of the classification model is correct, they have just not developed it along the lines suggested by studies of human problem solving and by the range of expert systems that have been constructed.

Finally, before I move on to consider the production rule formalism, I want to point out that the idea of non-classification methods could have been better developed also. The authors argue that R1 does a special kind of classification. But I think it is more useful to say that R1 does construction, not classification. The point has computational implications that reveal why EXPERT is not an adequate language for expressing R1's knowledge. Construction programs build some artifact or model whose components are related causally, spatially, or temporally. The multiple-disease diagnostic hypotheses of Caduceus are an example. Customization requirements of design or configuration problems generally entail this form of problem solving. With respect to language requirements, means must be provided for building up a partial solution that can be examined and modified. Independently-derived components can then be combined and/or used to constrain placement of other components. In this way, the parts of an R1 configuration can spatially affect placement of later parts. For an interpretation problem, such as speech understanding, the parts support (serve to explain) one another causally at the same or multiple levels of abstraction. Relatively simple knowledge representation schemes, as used in EXPERT, need not provide means for posting and examining partial solutions for they are not constructing hypotheses; they select solutions from a pre-specified list built into the program.

Weiss and Kulikowski deserve a lot of credit for using the classification model as a simple description of expert systems. They recognize that there is a distinction between knowledge content and structure and that problem solving in diverse domains shares a common structure. There is ample evidence for this model in knowledge representation research (*e.g.,* the structured abstraction in KRL) and cognitive science (*e.g.,* Schank's MOPS, Chi's physics problem-solving research at the Learning Research Development Center in Pittsburgh, and Norman and Rumelhart's schema-based models of understanding). It is perhaps asking too much for the first practical guide to synthesize all of this research.

On the other hand, the analysis of expert systems provided by Weiss and Kulikowski could have been better if it weren't so closely tied to the production rule formalism. Saying simply that "problem solving in R1 primarily involves the matching of production rule arguments" (p. 60) is not very insightful. Similarly, saying that DASD/DART gets information via actions of rules is simply describing a program, how if/then statements are used. Of better quality is the description of how a questioning strategy can be separated from rules relating findings and hypotheses. In essence, the analysis presented here falls short because the authors do not consistently provide a "knowledge level analysis." They do not clearly separate descriptions of knowledge encoding in production rules, from implementation- independent descriptions of terms, relations, and interpretation processes.

Similarly, Weiss and Kulikowski describe an EXPERT "context" as a screen for a set of rules. Yet, the example they give is something we see in EMYCIN and OPS5 rules as well: the procedural fact that hypothesis subtypes should not be considered if the hypothesis class is ruled out. Weiss and Kulikowski do not consider the advantages of expressing a subtype relation as a declarative fact and building into the tool (or providing primitives for controlling) how such a hierarchy should be interpreted. Thus, a principle is left implicit that could be usefully incorporated in other expert systems (Clancey, 1983). The concept of top-down refinement is not part of the framework, so its encoding becomes part of the folklore for using the tool, putting a premium on documentation and apprenticeship learning. Thus, knowledge engineering, ironically in its very attempt to capture what is learned by apprenticeship in various domains, fails to formalize the very principles it laboriously teaches to its own students.

The simplicity of a framework like EXPERT makes its syntax easy to learn. The price is paid in actually figuring out how to use EXPERT to encode knowledge and in the difficulty of modifying a knowledge base with implicit facts. Surely, the experienced knowledge engineer would benefit from saying what he means directly, rather than redundantly programming the same design throughout his program in every system he constructs. It appears that EXPERT has inherited a few of the features of FORTRAN, the language in which it is implemented.

If implicit knowledge were not enough—and I should say that EXPERT does far better than EMYCIN on this score—rule-based systems are notorious for using procedural semantics. For example, on page 116 you will find rules that conclude that various hypotheses are not present. Now everything you have learned so far indicates that this must be evidence against these hypotheses, right? No, this is simply a trick: you rule out intermediate conclusions (*e.g.,* classes of hypotheses) so they don't show up as "redundant" conclusions in the final output. The same games are played in EMYCIN, where to prevent an irrelevant question from being asked one marks it as already having been asked (a variant is to conclude that the answer of the question is "don't ask"). Programs like this have an inconsistent record of rules and conclusions that do not reflect what is true. Rather they can only be read and understood by knowing the specific procedure that will be interpreting them. This narrow view of what knowledge is blows explanation and tutoring systems out of the water!

If designing expert programs according to the tactics of obsolete programming languages is "practical," it can only be for the short run. A clear benefit of knowledge engineering will follow from building on shared libraries of knowledge. You can be sure that languages will be better structured and developed than EXPERT, EMYCIN, and OPS5. Implementation aside, Weiss and Kulikowski present the basic framework of the classification model in a simple, easy to understand way that I think will be useful for beginners. I only hope that more structured languages take hold before conversion of huge, rule-based systems becomes a major problem.

References

Chandrasekaran, B. (1984) Expert systems: Matching techniques to tasks. In W. Reitman (Ed), AI *Applications for Business.* Ablex Publishing Corp

Clancey, W J (1983) *The advantages of abstract control knowledge in expert system design AAAI-83*:74-78

Clancey, W. J (1984) *Classification problem solving AAAI-84*: 49-55.

Kulikowski, C A. (1980) Artificial Intelligence methods and systems for medical consultation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-2:464-476.

William J. Clancey
Heuristic Programming Project
Stanford University