Proceedings of ACM Annual Conference (pp. 11–14). New York: ACM.
Reprinted in A. Gupta and B. E. Prasad (Eds.), Expert Systems—Principles, IEEE Press, 1988.

KNOWLEDGE ACQUISITION FOR CLASSIFICATION EXPERT SYSTEMS

William J. Clancey
Heuristic Programming Project
Stanford University
Stanford, California

ABSTRACT

Expert systems are generally described by a mixture of terms that confuse implementation language with knowledge structure and the search process. This confusion makes it difficult to analyze new problems and to derive a set of knowledge engineering principles. A rigorous, logical description of expert systems reveals that a small set of terms and relations can be used to describe many rule-based expert systems. In particular, one common method for solving problems is by *classification*—heuristically relating data abstractions to a pre-enumerated network of solutions. This model can be used as a framework for knowledge acquisition, particularly in the early stages for organizing the expert's vocabulary and decomposing problems.

CR Categories and Subject Descriptors: I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods -- representations

I WHAT IS KNOWLEDGE ACQUISITION?

Knowledge acquisition is the general term used for the process of developing a computational problem-solving model, specifically a program to be used in some consultative or advisory role. Such programs are generally called "expert systems" [9]. The knowledge acquisition process actually has several steps. The most important are: selecting a problem to be solved by the program, interviewing an expert, codifying the knowledge in some representation language, and refining the knowledge base by testing it and extending its capability. A broad treatment of this topic would relate knowledge acquisition to learning and theory formation.

In this paper we are concerned with the early stage of knowledge acquisition. We advocate a general methodology for analyzing problems, called *knowledge level analysis*, that is independent of the implementation language that will be used to build the expert system. We introduce the *classification model* as one general framework which can be used to describe how an expert solves a problem. This framework can serve as a basis for selecting problems that are appropriate for a given representation language.

The reader will need to practice these ideas to fully understand them and to learn how to apply them effectively. A much longer, tutorial paper would be helpful for this purpose; this paper is only intended to be an introduction to the ideas.

II A KNOWLEDGE LEVEL ANALYSIS

Previous research has described expert systems almost exclusively in terms of representational terminology (e.g., "rules"), kinds of problems (e.g., "diagnosis" or "design"), or issues of search (e.g., "guessing is required") (for example, see [11]). A "knowledge level" analysis involves describing inference and search structure independently, separating issues of knowledge structure from representation and search [10, 16, 15].

The terminology of many representation languages is not adequate for describing knowledge structure and search. For example, consider the "white blood count rule" shown in Figure 1 (slightly restated for simplicity) from the Mycin program [18].

```
RULE557
If: 1) The infection is meningitis,
    2) The type of the infection is bacterial,
    3) No organism has been directly identified,
    4) Complete blood count results are available,
       and
    5) The white blood count is less than 2500

Then:
   The organisms that might be causing the
       infection are Pseudomonas (.5 certainty),
       Klebsiella (.5), and E.coli (.75).
```

Figure 1: Typical Mycin rule

At the implementation level, we say that this rule makes a conclusion about a goal in its action part ("What are the organisms causing infection?"). The program determines whether the premise part of the rule is satisfied by pursuing premise goals (e.g., "Is a complete blood count available?") by recursive invocation of its rule interpreter. This is the vocabulary of the Emycin language: rules, premise, action, goals, and backward chaining [19].

A *knowledge level description* of this rule would involve the following statements:

- *Terminological statements:* Meningitis, bacterial meningitis, and e.coli bacterial meningitis are *hypotheses*; the complete blood count and the white blood count are *findings*.
- *Relational statements:* Bacterial meningitis is *a kind of* meningitis infection; e.coli bacterial meningitis is a kind of bacterial meningitis; the white blood count is *a component of* a complete white count; a diminished white blood count *causes* e.coli bacterial meningitis infection.

- *Inferential statements:* If a finding is a component of another finding, conclude that the more specific finding is unavailable if the more general finding is unavailable; if a hypothesis is a kind of more general hypothesis, disbelieve the more specific hypothesis if the general hypothesis is disbelieved.

Thus, we have described the knowledge contained in this rule in terms of a set of terms (e.g., finding), relations (e.g., kind of), and inferential or search axioms describing how the knowledge is to be used in problem solving (e.g., a top-down refinement procedure). None of this terminology is provided by the Emycin representation language. *Instead, this knowledge is implicit in the rule.* It is common to say that the rule is "compiled" knowledge, though we must be clear that there are different kinds of "compilation" involved here:

- *Proceduralization*: Knowledge is not stated as explicit terms and relations, but as a procedure, a set of ordered things to do corresponding to the rule's clauses. The procedure is not stated explicitly, but is instantiated in the rule. This is the same meaning of knowledge compilation used in learning research [14].

- *Omission of causal model*: Intermediate causal steps may be omitted. The connection between white blood count and e.coli has been left out. The causal connection is heuristic, it may not always be true. Leaving out the intermediate steps makes search more efficient, a form of compilation. (Note that an expert may be using a heuristic precisely because what is left out, the causal mechanism, is not understood. Thus, no true compilation has occurred.)

- *Schematic knowledge*: The terms contained in this rule are experiential abstractions. Diseases are patterns of what has been seen before and how they are manifested. Thus, a "syndrome" is a compilation of how, in the problem-solver's experience, symptoms tend to occur together. This is to be contrasted with a structural model of body systems and their functions.

A knowledge level analysis has tremendous advantages. Perhaps the most important is a clear explanation of the basis of a rule. In fact, much of the work reported here has its origin in an effort to construct an instructional program from Mycin [4]. Given a knowledge level analysis, it is possible to devise a better representation language that allows us to make terms, relations, and search procedure explicit [3]. Moreover, the search procedure then becomes an object of study itself, so we can begin to uncover mathematical, sociological, and case population constraints from which the procedure is derived [5, 6].

In summary, the essence of a knowledge level analysis is making distinctions—finding patterns—and relating them to the computational process. We think in terms of input and output and how they are related. Rather than talking about goals, we talk about findings (input) and hypotheses (output). Rather than talking about rules, we talk about subtype and causality. We observe that some terms are abstractions of others. We make a distinction between a network of relations and the procedure by which it is interpreted. We point out that every heuristic rule has an implicit justification, just as every procedure is based on implicit ordering constraints.

III CLASSIFICATION EXPERT SYSTEMS

When we describe expert systems at the knowledge level, we find that the terms, relations, and search procedures themselves form a pattern. In particular, many expert systems solve problems by *classification.* Examples include programs that perform diagnosis, catalog selection, and even planning. The knowledge of these programs has a characteristic structure involving systematic relation of data to a previously known set of solutions by processes of data abstraction, heuristic association to a schema network, and refinement [7]. Mycin can be described in these terms, as shown in Figure 2 (the general structure is given, followed by an example inference path).

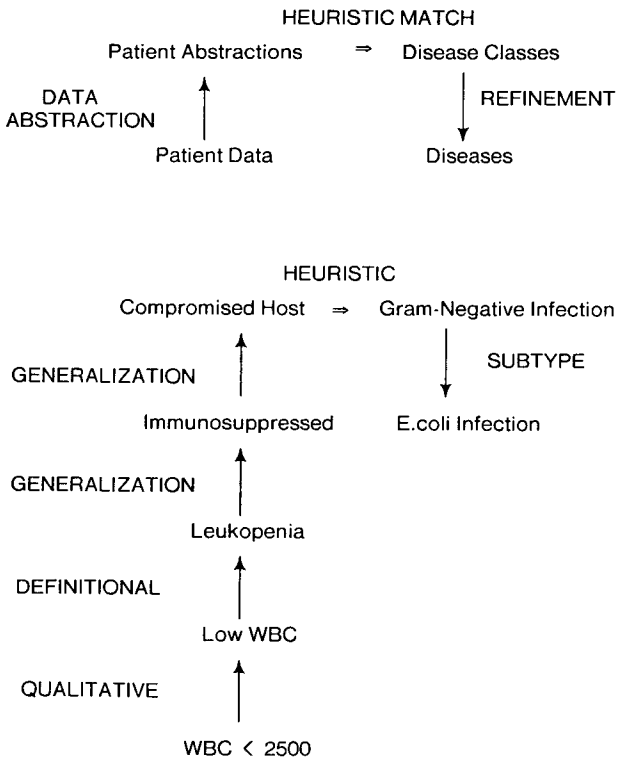

Figure 2: The inference structure of Mycin

The classification problem solving model describes how a particular problem solver solves a problem. It is not a description of a kind of problem. For example, medical diagnosis cannot always be accomplished by classification. Rather, whenever a problem solver *selects* a "canned" solution, he is solving the problem by classification. This is certainly a large part of what we mean by experiential knowledge: knowledge of problems and solutions and how they are related. Any problem may conceivably be solved by classification. We construct a program to solve problems this way by building in a list of possible solutions and the procedure for selecting among them. The essence of classification problem solving is that the solution to a problem is selected from a list of pre-enumerated possible solutions.

For completeness, we mention in passing that *construction* is the alternative to classification, and it involves piecing together a solution. A good example is the R1 program for computer system configuration. R1 [13] constructs a configuration from pieces. If the program merely selected a configuration from a library of possibilities, perhaps hierarchically arranged, it would be solving the problem by classification.

IV HEURISTICS FOR KNOWLEDGE ACQUISITION

The classification model provides a basis for interviewing an expert and structuring how he solves a problem. We can offer the following heuristics that follow from the model:

1. Problem selection

   - Look for problems that can be solved by classification: Are the solutions enumerable, stereotypic configurations, plans, diagnoses, etc.?

   - Decompose problems into sequences of classification problems, treat them separately, but work backwards from the final problem. For example, need/requirements analysis may be solved by classification, with a solution heuristically related to ultimate solutions (products, services, etc.)—work backwards from these ultimate solutions. Another common example: consider what kinds of repair you will offer before analyzing the diagnosis problem.

   - Early on, define the problem in terms of input and output and the kinds of relations. Try to distinguish between substances and processes. What is observable? Does causality play a role? For diagnosis, is there a disorder or abnormal state network? Is there a hierarchy of disorder processes (what can go wrong)?

2. Knowledge level analysis: a structured way for identifying terms and relations:

   - List all possible solutions the program may output; organize into classes and hierarchies if appropriate. Be clear about what the solutions are: plans, processes, configurations, etc. A confusion at this point may mean that there are separable problems. Be clear about *types*, that is, don't mix different kinds of things (e.g., descriptions of diseases with descriptions of people). All solutions should at a high level belong to a single class.

   - List classes of data that will be input to the program (no need to be exhaustive at this point, unless the list is under a few dozen items); will any data be numeric? Organize into classes and hierarchies to the extent possible.

   - Identify relations among the data: generalizations, definitions, and qualitative abstraction. Exact attention to relations is difficult, but essential to be sure the problem is properly decomposed. For example, take care to distinguish *an abstraction of data* according to definition and from *an abstraction of the solution* that is matched by direct identification of some features. (For example, "white blood count less than 2500" is the *definition* of leukopenia (a data abstraction); "gram-negative rod" *matches the features* of E.Coli (a diagnostic solution)). A common problem is that the expert will leave out qualitative abstractions, stating associations in terms of numeric data, or vice versa—not indicating until later that data are actually numeric.

   - Establish the heuristics that link data to solutions *after* establishing the network of solutions. To avoid identifying a solution as a datum, be aware that some rules may relate solutions to one another non-hierarchically (sometimes called a *complication*).

   - Treat the search process separately. It is essential to model the expert's inference structure (terms and relations), but not as important to model the search process he uses. For example, a program may use top-down refinement within a hierarchy of solutions, while the expert may use a more opportunistic, hypothesis-formation approach. Modeling search process is much more difficult, and is in general not necessary computationally for classification expert programs of the size constructed today. (Knowledge engineers implementing systems in backchaining rule systems usually make this simplification.)

3. Implementation: It is advantageous to use a programming language that allows relations to be made explicit, especially hierarchies. Top-down refinement can be easily encoded using clause ordering, but this approach leads to redundant, more complex rules, with a loss of explanation capability. Better engineering suggests separating the inference and process structure.

4. Refinement: Following the usual approach of improving the knowledge structure by testing the program on a variety of problems, the classification model suggests selecting cases that will test the program's ability to discriminate among solutions. One might begin with classic cases corresponding to each solution, then systematically pick problems with similar input, but different solutions.

This list obviously is not a formula for knowledge acquisition. Indeed, the order of knowledge level analysis given here (basically, a bottom-up, output-to-input approach) may be a useful organization for the learner, but the expert may find it difficult to directly describe what he knows in this way. It may be preferable to present problems to the expert and quiz him about what he is doing. Thus, the analysis presented here is a systematic framework for describing knowledge, not for eliciting it.

V RELATED RESEARCH

Most previous knowledge acquisition research has involved program aids for implementing an expert program in some language, such as the structured editors and prompters in Emycin [19] and Teiresias [8]. The knowledge elicitation process is discussed in some detail in [11].

The SEEK system by Politakis [17] performs systematic experiments for improving heuristics by refining the patterns of data that match solutions. Bennett's ROGET [1] uses a framework that bears familial resemblance to the classification model as an aid for problem selection and initial problem formulation. Boose's ETS program [2] uses a theory of how people make distinctions among concepts (called "personal construct theory," devised by Kelly) as a basis for an interview in which a solution hierarchy of patterns is constructed. Kuipers [12] describes a protocol analysis method for constructing a model of causal knowledge. He suggests using knowledge-level distinctions such as "substance" and "process."

VI CONCLUSIONS

We have seen that knowledge can be encoded in a representation, such as production rules, so that relations and search procedure are implicit. A logic-based analysis specifies terms and relations and separates inference from search procedure. Knowledge and search procedures for solving different problems can be compared at this level; one generic form of knowledge organization is called the classification model. The model suggests a set of heuristics for recognizing whether a problem can be solved by classification and for systematically acquiring the knowledge network.

Future research might follow Boose's example in applying psychology to the knowledge acquisition process. Studies of the nature of categorization in memory and learning are directly relevant. Clearly, there may be fruitful overlap between studies of expertise and principles for building expert systems. In a related form of empirical study, knowledge representation research may be sharply focused by deriving new languages from knowledge level analyses of existing expert systems.

ACKNOWLEDGMENTS

I would like to thank Diane Hasling, Mark Richer, and David Wilkins for their continuing contributions to the Neomycin/Guidon project. This research has been supported in part by ONR and ARI Contract N00014-79C-0302. Computational resources have been provided by the SUMEX-AIM facility (NIH grant RR00785).

References

1. Bennett, J. ROGET: A knowledge-based consultant for acquiring the conceptual structure of an expert system. HPP Memo 83-24, Stanford University, October, 1983.

2. Boose, J. Personal construct theory and the transfer of human expertise. Proceedings of the National Conference on AI, Austin, TX, Aug., 1984, pp. (in press).

3. Clancey, W. J. and Letsinger, R. NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching. Proceedings of the Seventh International Joint Conference on Artificial Intelligence, Aug., 1981, pp. 829-836. (Revised version to appear in Clancey and Shortliffe (editors), *Readings in Medical Artificial Intelligence: The First Decade*, Addison-Wesley, 1983)

4. Clancey, W. J. "The epistemology of a rule-based expert system: A framework for explanation." *Artificial Intelligence 20*, 3 (1983), 215-251.

5. Clancey, W. J. The advantages of abstract control knowledge in expert system design. Proceedings of the National Conference on AI, Washington, D.C., Aug., 1983, pp. 74-78.

6. Clancey, W. J. Acquiring, representing, and evaluating a competence model of diagnosis. HPP Memo 84-2, Stanford University, Feb., 1984. (To appear in Chi, Glaser, and Farr (Eds.), *The Nature of Expertise*, in preparation.)

7. Clancey, W. J. Classification problem solving. Proceedings of the National Conference on AI, Austin, TX, Aug., 1984, pp. (in press).

8. Davis, R. and Lenat, D.. *Knowledge-Based Systems in Artificial Intelligence*. McGraw Hill, New York, 1982.

9. Duda, R. O. and Shortliffe, E. H. "Expert systems research." *Science 220* (1983), 261-268.

10. Hayes, P.J. In defence of logic. Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Aug., 1977, pp. 559-565.

11. Hayes-Roth, F., Waterman, D., and Lenat, D. (eds.).. *Building expert systems*. Addison-Wesley, New York, 1983.

12. Kuipers B. and Kassirer, J. P. Causal reasoning in medicine: Analysis of a protocol. Working Paper, TUWPICS 20, Tufts University, February, 1983. (To appear in *Cognitive Science*)

13. McDermott, J. "R1: A rule-based configurer of computer systems." *Artificial Intelligence 19*, 1 (1982), 39-88.

14. Neves, D. M. and Anderson, J. R. Knowledge compilation: Mechanisms for the automization of cognitive skills. In *Cognitive Skills and their Acquisition*, Anderson, Ed.,Lawrence Erlbaum Associates, Hillsdale, 1981.

15. Newell, A. "The knowledge level." *Artificial Intelligence 18*, 1 (1982), 87-127.

16. Nilsson, N. J. "The interplay between theoretical and experimental methods in Artificial Intelligence." *Cognition and Brain Theory 4*, 1 (1981), 69-74.

17. Politakis, P. and Weiss, S. M. "Using empirical analysis to refine expert system knowledge bases." *Artificial Intelligence 22*, 1 (1984), 23-48.

18. Shortliffe, E. H.. *Computer-based medical consultations: MYCIN*. Elsevier, New York, 1976.

19. van Melle, W. A domain-independent production rule system for consultation programs. Proceedings of the Sixth International Joint Conference on Artificial Intelligence, Aug., 1979, pp. 923-925.