PROBLEM FEATURES THAT INFLUENCE THE DESIGN OF EXPERT SYSTEMS[1]

Paul J. Kline and Steven B. Dolins

Artificial Intelligence Laboratory
Computer Science Center
Texas Instruments Incorporated
Dallas, TX. 75266

ABSTRACT

An analysis was made of a set of design guidelines for expert systems. These guidelines relate problem characteristics to appropriate AI implementation techniques. The analysis indicates there are five general problem features that are important for the proper use of a wide variety of AI implementation techniques. By being aware of these problem features, knowledge engineers improve their chances of coming up with the right design for expert systems. Awareness of these problem features should also help knowledge engineers take full advantage of new AI techniques as they emerge.

I Introduction

The designer of an expert system has to make a number of choices about implementation techniques:

- What knowledge representation technique should be used?
- What problem-solving strategy should be employed?
- How should uncertainty be handled?

Making the right decisions greatly simplifies the development of an expert system and helps ensure its lasting usefulness. However, making the right decisions means choosing the AI implementation techniques that are most appropriate for the problem at hand. This can be difficult, as there is no readily available source of guidance about the appropriate use of AI techniques.

In an effort to address this problem, Kline & Dolins (1985) present a list of 47 *Probing Questions* that relate problem features to AI implementation techniques. Given access to an expert in a particular problem area, a knowledge engineer should be able to use the Probing Questions to determine which of nearly 100 AI techniques and implementation strategies are best suited to the problem.

The Probing Questions were developed by 1) collecting claims about the appropriate use of AI techniques from the published literature on expert systems, 2) circulating draft questions among experienced expert systems builders for their comments and additions, and 3) revising the draft questions in response to the nine sets of comments received. In level of detail and completeness, the questions go a step beyond what can be found in Stefik et al. (1983).

Knowledge engineers are encouraged to use the resulting Probing Questions to help design their expert systems. (Copies of the questions with supporting documentation are available from the authors). However, two problems can arise in using the current set of Probing Questions to design expert systems:

- Since there are 47 Probing Questions, there is a danger of "losing sight of the forest amid all of the trees." During the course of discussions between expert and knowledge engineer, problem features will emerge that should have substantial impact on the design of the expert system. Knowledge engineers need to recognize the significance of those features when they are mentioned and they need to explore them thoroughly with the expert to avoid misunderstandings.
- The Probing Questions are designed to verify the appropriateness of a target set of AI implementation techniques. If the problem in question cannot be solved using these techniques, then there is no obvious way to use the Probing Questions to help evaluate proposed solutions relying on techniques that are not part of the target set.

[1]This research was supported by the Air Force Systems Command, Rome Air Development Center, Griffiss AFB, New York 13441 under contract no. F30602-83-C-0123.

Analysis of the Probing Questions indicates there are five issues that are important for the proper use of a wide variety of AI implementation techniques:

- When does the information needed to solve problems become available? Information can be available at program-design time, data-input time, or problem-solution time.
- What kind of connection is there between evidence and hypotheses? There are many kinds of evidence, and different program designs are required to obtain the maximum amount of leverage from each kind.
- What counts as a solution, and how many are there likely to be? An expert system is entitled to stop and declare the problem solved under a variety of different circumstances.
- How accommodating is the program's environment? That is, how much assistance in the form of information or guidance can the program rely on without compromising its usefulness?
- How can we help ensure that the program will expend its effort wisely? It can be difficult to solve problems within the constraints imposed by computational resources.

If knowledge engineers have these problem features in mind during their discussions with experts they improve their chances of getting the information they need to make good implementation decisions. These problem features should also be useful in evaluating solution proposals that depend on the use of new AI techniques.

II Analyzing a Problem Domain

An examination of the Probing Questions shows 23% of those questions inquire about the expected arrival time of information, 34% inquire about the connection between evidence and hypotheses, 19% inquire about the definition of solutions, 19% inquire about the degree of accommodation provided the program's environment, and 15% inquire about good use of resources. (The same question can be counted in more than one category). While a substantial fraction, 19%, of the Probing Questions do not raise any of the five issues, we were unable to discern significant commonalities among the remaining questions. These statistics suggest that if knowledge engineers have thoroughly explored the five issues in their problem domains, then they should be able to answer most of the Probing Questions. In the process, they will be able to determine the potential applicability of many AI implementation techniques.

The following section discusses the Probing Question that best illustrates the importance of the expected arrival time of information. This is followed by brief summaries of a few of the other Probing Questions that are part of the 23% inquiring about that general issue. In all cases, Probing Questions are labeled with the numbers they have in Kline & Dolins (1985). Subsequent sections use the same format to illustrate the other four issues.

A. When Does Information Become Available?

If the knowledge that makes it possible to solve problems is available at program-design time, then it is generally possible to build that knowledge directly into the design of an expert system. In other cases, important information only becomes available at data-input time or after some progress has been made toward finding a solution. In these cases, a design must be found to take advantage of the information as it emerges. Knowledge engineers should appreciate that whenever they identify a crucial item of information that makes it possible to solve problems in a domain, they also need to establish the expected arrival time of that information.

This issue comes up in the Probing Question shown in Fig. 1, which determines whether constraint propagation techniques will be required to find the values of variables. If it is known at *program-design time* that the values of certain variables will always be provided as part of the input data, then forward chaining can be used to determine the values of the other variables. On the other hand, if it is necessary to wait until *data-input time* to discover which variables have known values, then more general constraint propagation techniques will have to be employed. The choice between forward chaining and constraint propagation hinges on the question of exactly when information will be available about the identities of the variables with known values.

Other implementation strategies for expert systems are appropriate when certain kinds of information are available at program-design time. For example (Probing Question 2.3.1), if at design time it is possible to anticipate the major areas of uncertainty and unreliability the program will face, then building redundancy into the design might help deal with the uncertainty (Buchanan & Shortliffe 1984, p. 684f).

Besides program-design time and data-input time, there are cases in which crucial information does not appear until a partial solution to a problem has been obtained. For example (Probing Question 2.6.2), opportunistic search strategies wait for "islands" of certainty to emerge and

Is it necessary to find values for a number of variables that take on numeric or boolean values?
and
Are there constraints among the variables that make it possible to use the known values of some of the variables to solve for other variables?
and
Does the identity of the variables whose values are known at the outset differ from problem to problem?

Yes, differ from problem to problem → *Constraint Propagation*
No, same variables known at outset → *Forward-Chaining Rules*

Figure 1: Probing Question 2.2.1

then use those islands to help interpret neighboring regions of greater uncertainty (Nii, Feigenbaum, Anton, & Rockmore 1982). While the islands of certainty are crucial to solving a problem, it is impossible to say where those islands will be found until a certain amount of progress has been made toward developing a solution.

B. What Kind of Connection is There Between Evidence and Hypotheses?

A wide variety of implementation strategies have been employed in expert systems in order to extract the maximum amount of leverage from evidence. The Probing Question in Fig. 2 is looking for several different kinds of connections between evidence and hypotheses.

As this question suggests, a test for detecting that a candidate is *not* a genuine solution can produce a positive conclusion by eliminating all but one of a set of candidates, i.e., confirmation by exclusion as in (Pople 1982, p. 130f.). The negative connection between evidence and hypotheses leads to a different expert system design than is obtained when there is total reliance on positive connections.

In other cases (Probing Question 2.4.9), a particular piece of evidence restricts the solution to a range of possibilities without saying anything about which of those possibilities is actually the right one—for example, Pople's "constrictors" (1977, p. 1033). This kind of connection between evidence and hypotheses leads naturally to expert systems that organize hypotheses into a hierarchy and proceed from general hypotheses (e.g., lung disease) to more specific hypotheses (e.g., emphysema).

As a final example, it has been observed (Clancey 1984, pp. 550f; Kahn 1984, p. 25f) that the nature of the connection between evidence and hypotheses influences the choice of "shallow" versus "deep" reasoning in expert systems (Probing Question 2.1.1). In some diagnosis problems, the evidence is linked directly to bottom-line conclusions. Shallow reasoning employing heuristic associations is appropriate for these problems. In other diagnosis problems, evidence can be found that also confirms intermediate steps along a causal path connecting ultimate causes to symptoms. Deep reasoning employing a model of the operative causal relationships may be appropriate for these sorts of problems.

C. What Counts as a Solution, and How Many are There Likely to Be?

Knowledge engineers and experts need to achieve a clear understanding about the circumstances that will entitle the program to stop and declare the problem is solved. Different stopping criteria will be appropriate in different problem domains. For example, the output of the R1 expert system (McDermott 1982) is a functionally acceptable VAX system configuration. There is no guarantee the optimal configuration is found, but R1 is quite useful nonetheless. In other problem domains a program would need to keep working until an optimal solution is found.

A closely related question is the number of solutions that are expected. For example, some medical expert systems can safely assume patients will only have one of the diseases the program is capable of diagnosing (e.g., the expert system discussed in Reggia, Nau, and Wang, 1984, for determining the cause of the wasting of the muscles of the lower legs). Other medical expert systems must be prepared to deal with patients with multiple diseases. The Probing Question in Fig. 3 makes recommendations for the case where it is not reasonable to stop the diagnostic process as soon as one solution is found.

Discussions of the subtractive method recommended in Probing Question 2.2.5 and examples of heuristics for judging parsimony and detecting competitors can be found

Is it possible to construct a test that can be applied to each candidate solution, so that passing the test proves the candidate is a genuine solution?
(*e.g., A combination is clearly the right one if it opens the safe*)
or
Is it possible to construct a test, so that failing the test proves the candidate is not a genuine solution?
(*e.g., Blood tests can rule out paternity, but not establish it.*)
or
Is there only a large "gray area" of better and worse candidates to choose from?

Rule candidates in, and there is an efficient generator	→ *Generate-And-Test*
Rule candidates out	→ *Generate-And-Test*, *Pruning*, or *Confirmation By Exclusion*
Gray area	→ *Scoring Functions*, *Group and Differentiate*, *Opportunistic Search*, etc.

Figure 2: Probing Question 2.7.3

in Pople (1977 p. 1032), Reggia, Nau, and Wang (1984), and Patil, Szolovits & Schwartz (1981).

There is a continuum of expert system problems that ranges from multiple solutions at one extreme, passes through a point where there is exactly one legitimate solution, and finally reaches a point at the other extreme where there are no solutions to the problem as originally stated. ISIS (Fox, 1983) provides an illustration of the "no solutions" end of this continuum. ISIS attempts to construct job-shop schedules that satisfy a number of constraints. It often turns out that there are implicit conflicts between the constraints making it impossible to find any schedule that satisfies them all. ISIS employs a constraint relaxation scheme to define new problems that have a better chance of being successfully solved (Probing Question 2.9.2).

Sensor interpretation problems will often be examples of the "one solution" point on this continuum. If we can assume there is some true state of the world that gives rise to the sensor data, then, in principle, there is only one legitimate solution. If the sensor data is sufficiently rich to uniquely determine the underlying state of the world (Probing Question 2.3.5), then an interpretation expert system can be satisfied with finding one solution (Feigenbaum 1977, p. 1025).

D. How Accommodating is the Program's Environment?

Expert systems provide assistance to their users, but in order to do so, the programs themselves generally need assistance in the form of information or guidance. The amount of assistance that a program can rely on without compromising its usefulness is another problem feature that influences the design of expert systems. The Probing Question in Fig. 4 is concerned with this issue.

This question suggests that one extreme of accommo-

Is this a diagnosis problem?
and
Would it be unwise to assume that there is only a single underlying fault because multiple faults are either too common or too serious to run the risk of a mis-diagnosis?

Yes → Solve a sequence of problems that "Subtract Off" previously accounted for manifestations. The system will need to use heuristic criteria to determine the most parsimonious explanation and also to distinguish between competing and complementary hypotheses.

Figure 3: Probing Question 2.2.5

What is the nature of the environment that provides inputs to the program?

1. **Cooperative and knowledgeable users will provide inputs.**
2. **Users are cooperative, but not always knowledgeable. That is, certain users are likely to give unreliable answers to questions posed by the system.**
3. **The environment is hostile and might therefore try to mislead the program with false inputs.** *(e.g., Enemy ships might try to hide by emitting misleading signals or no signals at all.)*
4. **Neutral environment that is a source of data, but does not try to influence the program one way or another.**

Both cooperative and knowledgeable	→ Accept the information that is input as accurate and complete.
Not always knowledgeable	→ Tailor information gathering to the knowledge level of individual user, or allow the users to indicate how certain they are that their answers are correct, or apply more consistency checks when users are less knowledgeable.
Hostile	→ Expend much effort in *Consistency Checking*, set up demons to look for evidence of deception, *Reason Explicitly About Uncertainty*, use *Endorsement-Based Approaches* to try to resolve uncertainties, etc.
Neutral	→ Expend moderate effort on *Consistency Checking*.

Figure 4: Probing Question 2.2.6

dation is the misleading information arising from deception in military settings and the other extreme of accommodation is the reliable information provided by cooperative and knowledgeable users. However, if guidance rather than information is at issue (Probing Question 2.9.1), then an extreme case of accommodation is a division of labor between man and machine where the user makes all the critical decisions. The program might display the decision options and then trace the consequences of the user's decisions. With this kind of arrangement, the program is not capable of solving the entire problem by itself and is dependent on guidance from a very accommodating environment; i.e., a competent user to whom it can defer decisions.

E. How Can We Ensure that Effort is Expended Wisely?

Many expert systems operate in domains where a combinatorial explosion of possibilities will defeat them if they do not expend their efforts intelligently. The Probing Question in Fig. 5 illustrates a variety of approaches to deciding what to do next so problems get solved within the constraints imposed by resource limitations.

This question contrasts various global strategies for directing the problem solving process. However, it is possible to make finer distinctions within some of the broad categories that Probing Question 2.9.4 treats as equivalent. For example, the choice of a general purpose search strategy also has implications for the program's ability to make good use of its resources. The data-driven reasoning provided by forward-chaining search allows the program to immediately recognize the implications of evidence that strongly suggests a particular hypothesis (Probing Question 2.6.4). This rapid appreciation of the consequences of new information is important in some problems. In other problems, the goal-driven reasoning provided by backward-chaining search leads to a better expenditure of resources. This will be the case if it is important to be sure that all inferences made could help achieve the program's current goals.

III Analyzing an AI Technique

The Probing Questions provide guidance about the appropriate use of nearly 100 AI techniques and implementation strategies. However, there are other AI techniques not included in that set, and new techniques are

Is there a fixed order of subtasks that solves most problems in this domain?
or
Are the potential lines of reasoning few enough that the program can afford to investigate them in the order that is most convenient for the reasoning strategy?
or
Do experts routinely use their knowledge of the domain to make good choices of subproblems to work on next?
or
Will the program need to estimate the costs and benefits expected from invoking a line of reasoning so as to best allocate computational resources among a wide range of possible lines of reasoning?

Fixed order of tasks →	Hard-wire the flow of control, e.g., conventional programming or the *Match* strategy.
At the convenience of the reasoning strategy →	General-purpose search strategies, i.e., forward-chaining, backward-chaining, etc.
Domain knowledge determines next task →	Use *Control-Rules* that embody domain knowledge to reorder the *Agenda*, set focus tasks, or invoke rule sets.
Estimate costs and benefits →	Devise an *Intelligent Scheduler* to order tasks on an *Agenda* according to their expected benefits. *Meta-Knowledge* is required about the costs and benefits associated with potential lines of reasoning.

Figure 5: Probing Question 2.9.4

being developed fairly rapidly. Since descriptions of an AI technique often do not say what problem features make that technique appropriate, it would be helpful to identify questions knowledge engineers could ask in order to make a decision for themselves.

The five issues discussed in this paper provide some candidate questions. Given a new AI technique, a knowledge engineer should ask: 1) what assumptions the new technique makes about the arrival time of information, 2) what kind of connection between evidence and hypotheses that technique assumes, 3) what assumptions this technique makes about the nature of solutions, 4)how accommodating the program's environment would have to be for the technique to be useful, and 5) whether that technique helps the program expend its effort wisely.

One way to estimate how useful these questions are likely to be in characterizing the use of a new AI technique is to determine how often they help characterize techniques in the current collection. It was found that these questions helped characterize the appropriate use of approximately two thirds of those techniques.

As a concrete example, a case was discovered where more precision was needed in asking about the expected arrival time of information. A Probing Question that determines if means-ends analysis is an appropriate search strategy asks:

> **Is it relatively easy to guess that a certain crucial step will be required to solve the problem?**

Given the previous discussion of the need to distinguish program-design time from data-input time and problem-solution time, it is now clear this question should have been phrased

> **Is it relatively easy to guess *at program-design time* that a certain crucial step will be required to solve the problem?**

By paying attention to the issues discussed in this paper, it should be possible to avoid this kind of mistake in future analyses of AI techniques.

IV Conclusions

Analysis of the 47 Probing Questions in Kline & Dolins (1985) indicates there are five issues that are important for the proper use of a wide variety of AI implementation

techniques. By being aware of these issues, knowledge engineers improve their chances of coming up with the right design for expert systems. Awareness of these issues should also help knowledge engineers take advantage of new AI techniques as they emerge.

ACKNOWLEDGEMENTS

We would like to thank Dr. Northrup Fowler III of Rome Air Development Center for his valuable suggestions during the course of this research. We are very grateful to the expert systems builders who were kind enough to comment on draft versions of the Probing Questions: Bruce Buchanan, Ruven Brooks, John Kunz, Penny Nii, Michael Genesereth, Bruce Porter, Robert Drazovich, Robert Neches, Tim Finin, Barbara Hayes-Roth, Casimir Kulikowski, and Jim Kornell.

REFERENCES

Buchanan, B.G. & Shortliffe, E.H. (Eds.), *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Reading, MA: Addison-Wesley, 1984.

Clancey, W.J. Extensions to rules for explanation and tutoring. *Artificial Intelligence*, 1983, *20*, pp. 215–251.

Fox, M.S. *Constraint-directed search: A case study of job-shop scheduling.* Ph.D. dissertation, Computer Science Department, CMU, Pittsburgh, PA, 1983.

Feigenbaum, E.A. The art of artificial intelligence: I. Themes and case studies of knowledge engineering. *Proceedings of IJCAI-77*, pp. 1014–1049.

Kahn, G. On when diagnostic systems want to do without causal knowledge. In *ECAI-84: Advances in Artificial Intelligence.* T. O'Shea (Ed.), Elsevier, 1984, pp. 21–30.

Kline, P.J. & Dolins, S.B. *Choosing Architectures for Expert Systems.* Final Technical Report RADC-TR-85-192, October 1985, Rome Air Development Center, Griffiss AFB, NY, 13441. (Available through the National Technical Information Service or the authors.)

McDermott, J. R1: A Rule-Based Configurer of Computer Systems. *Artificial Intelligence*, 1982, *19*, pp. 39–88.

Nii, H.P., Feigenbaum, E.A., Anton, J.J. & Rockmore, A.J. Signal-to-symbol transformation: HASP/SIAP case study. *The AI Magazine*, Volume III, No. 2, Spring 1982.

Patil, R.S., Szolovits, P., Schwartz, W.B. Causal understanding of patient illness in medical diagnosis. *Proceedings of IJCAI-81*, pp. 893–899.

Pople, H.E.Jr. The formation of composite hypotheses in diagnostic problem solving: An exercise in synthetic reasoning. *Proceedings of IJCAI-77*, pp. 1030–1037.

Pople, H.E.Jr. Heuristic methods for imposing structure on ill-structured problems: The structuring of medical diagnostics. In *Artificial Intelligence in Medicine.* P. Szolovits (Ed.), Boulder, CO: Westview Press, 1982.

Reggia, J.A., Nau, D.S., and Wang, P.Y. Diagnostic expert systems based on a set covering method. In M.J. Coombs, (Ed.) *Developments in expert systems*, New York: Academic Press, 1984.

Stefik, M., Aikins, J., Balzer, R., Benoit, J., Birnbaum, L., Hayes-Roth, F. & Sacerdoti, E. The architecture of expert systems. In F. Hayes-Roth, D.A. Waterman, & D.B. Lenat (Eds.) *Building Expert Systems*, Reading, MA: Addison-Wesley, 1983.