

Materials:

1. Projectable of MYCIN dialog from Winston (p. 193 f)
2. Projectable of Luger p. 278
3. Projectable of Luger example car starting system - rules on pp. 287-288; and/or tree of rules Figure 8.8 p. 290
4. Projectable of Winston p. 186 (top)
5. Projectable of Winston IDENTIFIER rules and handout/demo of Prolog implementation
6. Projectable of Winston BAGGER rules and handout/demo of Prolog implementation

Introduction

A. We are now making something of a shift in our course - from looking at broad general principles (e.g. knowledge representation and search) to looking at specific AI applications. We will spend about 4 weeks looking at various areas, of which the topic for today is the first.

B. Recall the distinction between "strong" and "weak" methods of AI problem solving. What is it?

ASK

A weak method tackles a problem using a general search strategy - such as GPS. A strong method uses heuristic knowledge of the problem to produce a more informed search.

C. Expert systems rose to prominence in AI in the 1980's. An expert system is always focussed on problem-solving in a specific, narrow domain, and tries to capture the expertise of a human expert in that domain - typically in the form of "if-then" rules.

Expert systems arose in the context of symbolic AI, and are by and large an application of symbolic AI methods, but non-symbolic methods have sometimes been used in them as well.

D. We will consider three questions

1. What is an expert system?
2. Where are expert systems useful?
3. How is an expert system organized?

E. We will also look at a couple of very simple examples, including a Prolog implementation of each.

I. What is an Expert System?

- ---- - - - - - - - - - - - - - - - -

A. Historically, expert systems have been one of the most commercially successful applications of AI. Indeed, in the popular press AI was, at one time, largely equated with expert systems though, as we have seen, the field of AI is really much broader than that, and there have been numerous other areas of application of AI techniques, especially in recent years.

- B. An expert system is a system that embodies a significant portion of the specialized knowledge of a human expert in a specific, narrow domain.
1. Work in this area is based on the premise that what makes a person an expert is years of experience that enable him/her to recognize certain patterns in a problem as being similar to patterns he/she has seen previously. As a result, an expert can bring experience with the previous problem to bear on the current one.
 2. Expert systems attempt to codify this past experience in the form of rules of the form
 - if the current problem has this characteristic
 - then proceed along these lines
 3. A distinctive feature of expert systems is that this knowledge is stored in a very open and flexible way.
 - a. Most expert systems allow their users to inspect their rules in order to see how a particular conclusion was reached.
 - b. Many incorporate provisions for adding to or modifying the rules as experience with using the system suggests the need for refinement.
 4. Contrast this with traditional software packages.
 - a. Traditional software packages are usually distributed in object code form only; and their inner workings are sometimes regarded as a proprietary secret of their producers.
 - b. Modifying a traditional software packages (even trivially) requires production of a new release by the original supplier.
- C. The term "expert system" is itself controversial.
1. Some authors avoid the phrase "expert system", but instead speak of "rule-based systems". The latter term is perhaps preferable since, as authors like Winston point out, a so-called expert system is by no means fully equivalent to a human expert in the field.
 2. Other writers have called these systems "knowledge-based" systems.
 3. You will shortly read chapter one of a book by the brothers Hubert and Stuart Dreyfus. In this book, entitled *Mind Over Machine, The Power of Human Intuition and Experience in the Era of the Computer* they argue against the term "expert system". We will consider soon the reasons they put forth for not liking to use this term.
 4. Nonetheless, since the term "expert system" is the one that is commonly used in the literature (both popular and technical), we will use it too.
- D. Expert systems stand at the opposite extreme, methodologically, from approaches like GPS:

1. GPS - General problem solver - attempted to develop a paradigm which could tackle any problem. Had this approach worked, the same program would have been able to handle essentially any problem given it. (But it didn't work; GPS is, as Winston put it, "not a modern control structure". That particular line of work has hit a dead end.)
2. Expert systems, on the other hand, are highly domain-specific. A different expert system must be created for each specific problem; no two are totally alike. (However, there are a number of general principles and software tools which provide a lot of commonality.)

II. Where are Expert Systems Useful?

-- -----

A. Expert systems fall into two broad categories:

1. Synthetic systems produce something - often a plan - by putting together the pieces supplied as input data. For example, the XCON system that we talked about in connection with production systems was such a system.
2. Analytic systems reach conclusions as to the nature of some real world entity by examining various possible hypotheses to see which one(s) fit the input data. One example of such a system that is quite important historically was MYCIN.
 - a. The traditional method for diagnosing bacterial infections is via cultures. However, if a patient is seriously ill, there may not be time to wait for the result of a culture before starting treatment.
 - b. Specialists can often make a good "guess" on the basis of incomplete data about the bacterium, the site of the infection, symptoms etc. But such specialists are in short supply and one may not be readily available to diagnose a severely ill patient.
 - c. MYCIN was a conversational system that asked specific questions of the physician. Because the doctor may not be able to give an absolute answer to certain questions, he/she was allowed to include a degree of certainty in the answer which MYCIN would take into consideration in its reasoning.
 - d. Ultimately, MYCIN formed hypotheses as to possible infections (often more than one) and suggested a course of treatment with antibiotics that will "cover the bases."
Sample dialogue: PROJECT - Winston p. 193 f.
 - e. MYCIN itself was never actually put into use, because of legal and ethical questions surrounding the matter of using computers in medicine. However, it evidently outperformed members of the medical school faculty at Stanford - where it was developed - though it was not as accurate as an expert in the field. (MYCIN gave the correct diagnosis about 65% of the time; human experts averaged 80%).
3. Most expert systems fall in the analytic category.

- B. Luger gives a list of typical expert systems applications (originally taken from a book by Waterman)

PROJECT list from Luger p. 278

- C. He also give seven characteristics of a problem that is suitable for solution by an expert system. (pae 281)

1. The need for the solution justifies the cost and effort of building an expert system (typically several person-years of effort.)
2. Human expertise is not available in all situations where it is needed, due to:
 - a. A shortage of people with the needed expertise.
-- and/or
 - b. Expertise being needed at geographically remote sites, requiring a human expert to do extensive travelling.
3. The problems may be solved using symbolic reasoning techniques.
4. The problem domain is well-structured and does not require commonsense reasoning.
5. The problem may not be solved using traditional (algorithmic) computing methods.
6. Cooperative and articulate experts exist. This is a key point we will say more about in a moment.
7. The problem is of proper size and scope.
 - a. The problem must be small enough to allow a working system to be developed in reasonable time. This may require a narrow focussing of the objectives initially.
 - b. Often, a working system can be expanded as it is in use. (Many systems have been.) Note that the design for modifiability that characterizes these systems helps facilitate this.

- D. Two kinds of people play a key role in developing an expert system; a successful project requires the availability of both.

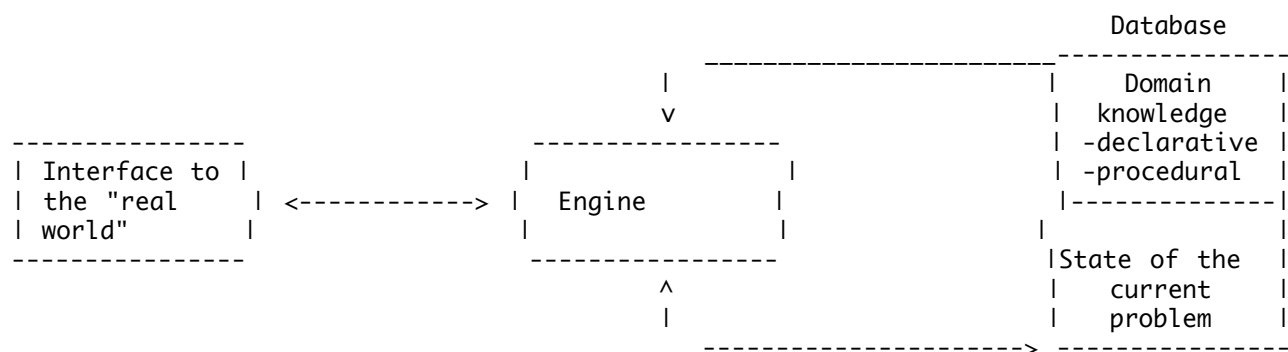
1. The DOMAIN EXPERT (or experts) provides the knowledge for the system.
 - a. This person/these person need not know anything about expert systems - and generally do not.
 - b. But they must be willing to cooperate in having their knowledge transferred to such a system. Obviously, this must be someone who is no threatened by the thought of "being replaced by a computer".
 - c. Building an expert system requires quite a time commitment on this person's/people's part, so they must be willing and able to give the time.

2. The KNOWLEDGE ENGINEER

- a. Though this person must be familiar with expert systems, he/she need not know much about the problem domain before the project starts. In fact, in some respects unfamiliarity with the problem domain is an asset, since it forces him/her to ask very basic questions - the sorts of things that will need to be built into the system.
 - b. The knowledge engineer interviews and questions the domain expert, and seeks to translate the expert's knowledge into a knowledge base of rules.
3. In addition to these two, PROSPECTIVE USERS must also be involved with the development to be sure that they can easily use the system. (This involves interface issues plus the kinds of dialogue they can have with the system.)
4. The development of an expert system is an iterative process.
- a. Rapid-prototyping techniques are typically used to get a preliminary version working early.
 - b. The developing system is tested in two ways:
 - i. Comparison of system output with correct answers given by the expert.
 - ii. Use by actual users under controlled conditions.
 - c. A typical system will undergo MANY revisions before it is ready to be put to actual use - and will continue to be revised over the years as it is used.

III. How is an Expert System Organized?

A. A general paradigm for this type of software system is as follows:



- 1. The engine is, as we shall see, relatively simple - though there are a number of different "styles" to choose from depending on the nature of the particular system.
- 2. The state of the current problem contains data about a particular problem that has either been provided as input or inferred by the program.

- a. For some expert systems, the input data is all provided up front, and the system proceeds from there. This is notably true in the case of systems that do synthesis, though it can be true in analytic systems as well.
 - b. For other systems, the program will periodically ask the user questions relevant to the hypothesis it is currently working on. This is more common in the case of systems that do analysis.
2. The program's knowledge resides in the domain-specific-knowledge portion of the database in the form of condition-action or if-then rules. Since these are the heart of the expert system, we will discuss them in some detail.
 3. This basic structure is often augmented by two additional components.
 - a. An explanation facility allows the user to ask questions during an actual consultation, like
 - i. How did you reach this conclusion?
 - ii. Why are you asking me this question?

(Both of these can be answered by the system by displaying part of a rule, as we will see later.)
 - b. A knowledge editor facilitates modifying and adding rules.
 4. What makes one expert system different from another is the content of the knowledge base - the basic program architecture may well be the same. This has led to the development of commercial EXPERT SYSTEM SHELLS, which are complete expert systems with an empty knowledge base, needing only appropriate rules (and some tailoring of the user interface) to be complete systems.
- B. Condition-action rules form the heart of an expert system's expertise. They generally attempt to capture heuristics, or rules of thumb, that an expert human problem solver might use in tackling a problem. For example, any automobile mechanic knows that if attempting to crank a car's engine results in the engine barely turning over, then a check of the condition of the battery is in order.
1. This might be captured by a rule of the form:


```

      if      engine cranks slowly
      then   check battery condition
      
```
 2. Notice that a heuristic like this is not infallible.
 - a. The same symptom might also be caused by a loose connection on the battery or starter, a defective starter motor etc.
 - b. But it represents an avenue to solving the problem that leads to success in a large number of cases. (Of course, an expert system for automobile starting problems might include additional rules to cover these possibilities too.)

3. Contrast this heuristic approach with an analytic approach, which might attempt to deduce the cause of the problem by studying the car's blueprints and systematically testing every component involved in the problem.
 - a. The latter approach might eventually come up with the correct solution, but only after possibly tearing the engine apart to check the condition of the bearings etc!
 - b. In industry, technicians are often much better than engineers at diagnosing problems with equipment. The latter tend to try to deduce what the problem must be by reasoning from the equipment's design, while the latter rely on past experience with similar problems. It is the "technician" approach that the expert system attempts to emulate.
 - c. Another way of putting this is that expert systems focus on what the PROBLEM SOLVER (the expert) does, rather than on the PROBLEM itself.
 - i. What makes this a strong way to solve problems is the expert knowledge that is captured by the rules. Expert systems have succeeded in many places where more general approaches have not.
 - ii. A key question is, how far can they go? An expert system is typically limited to a very narrow domain, and is of no use outside that domain.

Example: one wag has noted that if you asked MYCIN to help you with a problem starting your car, it would gladly prescribe an antibiotic for your carbeuretor :-)

4. Every rule has two parts:
 - a. The antecedents: one or more conditions that must hold for the rule to be applicable. If there are multiple antecedents, then all must hold.
 - i. If all the antecedent's of a given rule are true, then the rule is said to be TRIGGERED.
 - ii. However, an expert system only applies one rule at a time. This is called FIRING the rule. If multiple rules are triggered, then some strategy must be used for deciding which one to fire.
 - b. The consequents: one or more things to do when the rule is fired.
 - i. If the rule is part of a system for synthesis, a consequent may be actual physical steps to carry out, or to plan to carry out.
 - ii. If the rule is part of a system for analysis, a consequent will probably be an additional fact to believe - e.g. MYCIN concluding that certain symptoms are best explained by a particular organism.

iii. In general, a given rule may have one or many consequents. However, multiple consequents are more common in a system for synthesis, for a reason which will become apparent when we consider the control structure of an expert system.

c. That is, the knowledge base + engine of an expert system, together, constitute a form of production system.

C. The rule structure of a program, overall, forms an AND-OR tree.

1. Each rule, in itself, is an AND node, since all antecedents must hold before the consequent can be applied.
2. When (as is often the case) multiple rules lead to the same consequent, then we have an OR node: the consequent is accepted if any of the rules containing it fire.

Example: Luger gives an example of a very simple system for diagnosing car starting problems.

PROJECT Luger pp. 287-288

These rules form an AND/OR: PROJECT Luger Fig 8.8 p. 290

3. AND/OR trees facilitate answering questions about the program's reasoning. This is important, since the user might reasonably want to know how the program reached its conclusion, rather than just taking it on faith. (This is particularly true with systems like MYCIN, where the human doctor using it is still legally liable for what he prescribes for the patient.)

a. "How did you reach this conclusion?" questions can be answered by finding the rule that fired and established the consequent in question. The antecedents of that rule become the answer to the question. (Of course, if the consequent is established by two or more rules that have fired, then the program can give two separate lines of reasoning that converged on the conclusion.)

Example: if the system concludes that the problem with the car is the spark plugs, and is asked how it reached this conclusion, it would respond with antecedents of rule 1: "the engine is getting gas and it is not turning over". Further "how did you conclude this" questioning would lead back to the input the user supplied - the engine turns over, there is gas in the fuel tank, and there is gas in the carburetor.
[I.e. we work down the tree from the consequent we are trying to explain]

b. In an expert system that asks the user to respond to specific questions as it is running the user may ask "Why do you want to know?". The program can answer by giving the consequent of the rule it is attempting to apply. The answer might take the form "because I am exploring the hypothesis that <consequent>."

Example: if the car-starting system asks the user if there is gas in fuel tank, and the user asks "why do you want to know?", the system could answer "because I am exploring the

hypothesis that the engine is getting gas". Further "why do you want to know?" questions would lead back to the consequent of rule 1: "Because I am exploring the hypothesis that the problem is the spark plugs".

Example: PROJECT - Winston p. 186 (top)

Note: the ability to answer "why do you want to know?" questions is especially important in medical systems, because the tests needed to establish some piece of information may be expensive and or dangerous, and the human doctor might want to be sure the program has an adequate reason for requesting this before ordering the test(s). However, similar issues may arise in any situation where the system may require information that is difficult or expensive to obtain.

D. Rules are often classified into groups.

1. In a synthetic system, the groupings may correspond to major stages in the process. That is, the groupings may be based on the antecedents of the rule - those having antecedents appropriate when the process is just starting, those having antecedents that include the completion of the first major step etc.
2. In an analytic system, the groupings may be based on consequents.
 - a. Again, the grouping may correspond to stages in the analysis.
 - b. There might also be a grouping of rules that lead to similar conclusions. If the program explores as a hypothesis the consequence of one of the rules in a group, and if that rule fails, then depending on the reason for failure the program may either:
 - i. Explore a different rule in the same group - if the failure was caused by an antecedent that is not common to most other rules in the group.
 - ii. Reject all rules in the group - if the failure was caused by an antecedent that is common to all members of the group.
3. The system may be then structured to progress through the rules one group at a time, so that at any given time only a subset of the rules are candidates for triggering. This has the practical effect of improving efficiency by reducing the number of candidates the engine has to explore when a new fact is learned, and also makes the rule base more modular and thus easier to maintain.

E. The engine an expert system may use either forward backward chaining.

1. A forward chaining system is natural in terms of the structure of the rules.
 - a. The engine determines which rules are triggered by comparing their antecedents to the system state.
 - b. This approach enables us to infer ALL of the consequents that can be derived from the given data.

- i. For synthetic systems, this is certainly appropriate.
 - ii. For analytic systems, this may result in an undesired explosion of information.
- 2. Backward chaining systems are often useful for analysis.
 - a. Such systems may get information from the user by asking specific questions, as MYCIN does. It is desirable for the system to focus its question-asking, rather than asking every conceivable question. This focus arises by asking questions that relate to a common goal at the same time.
 - b. The engine may begin by selecting one possible overall conclusion (root of a subtree) as a hypothesis to investigate. Each of the antecedents of the rule leading to the conclusion then becomes a new hypothesis to check.
 - i. In checking a hypothesis, the engine looks for a rule that has the hypothesis as its consequent. If no such rule exists, it asks the user. (Under some circumstances, a rule may specify that the system should first ask the user, then try to infer the answer if he doesn't know it.)
 - ii. If one of the necessary antecedents of the original hypothesis is found to be untrue, the engine shifts to a new hypothesis. By now, though, it has gotten some information from the user, so it may be able to rule out certain hypotheses without asking for further information. [That is, when the system asks the user a question it saves the answer, so that it doesn't have to ask the same question again.]
 - iii. The search continues until one hypothesis is shown to be true or until all hypotheses are rejected.
 - iv. Note that this, in effect, amounts to a depth-first search of the rule graph, starting from conclusions and working toward given facts.
 - c. Backward chaining facilitates answering questions like "why are you asking me this?". The system can respond by giving the hypothesis it is exploring. (Of course, the system can answer the other kinds of questions we discussed above as before.)
 - d. Backward chaining is, as we know, the basic control regimen of Prolog. This makes Prolog a nice tool for building such systems. However, as we shall see shortly, Prolog can also be used for forward-chaining systems, with a bit more work.

IV. Two Examples

-- --- -----

- A. Both of these examples are taken from Patrick Henry Winston's book. In each case, we will first look at the rules, and then at a Prolog program that implements them.
- B. An analytic system - "Animal identifier"
 - 1. PROJECT Rules - Winston p. 177-180
 - 2. HANDOUT/DEMO Prolog code: identify(zelda). Note single consequent rule
- C. A synthetic system - a "Grocery Bagger"
 - 1. PROJECT Rules - Winston p. 169-174
 - 2. HANDOUT/DEMO Prolog code: [bagger, bagtest], do_test(test_order_63).

V. Alternatives to Rule-Based Expert Systems

- ----- -- ----- ----- -----

- A. The discussion thus far has focussed on the most common type of expert system - a rule-based system. (In fact, as we noted at the outset, some writers prefer the term "rule-based system" over "expert system").
- B. There are two other broad approaches to building systems that incorporate expert knowledge of a domain. We will not discuss these in depth, beyond noting what they are. Both were discussed in the book:

ASK

- 1. Case-Based Systems. A case-based system maintains a database of previously solved problems. To solve a new problem, the database is searched for similar problems, and the solution(s) to those is/are adapted to the new problem. (The search for a matching case is non-trivial!)

In fact, in some domains human experts seem to work precisely this way - e.g. law, medicine. [In these domains rule-based systems can be regarded as basically coding "case knowledge" in a more directly-applicable form.]

- 2. Model-Based Systems. A model-based system is built around theoretical knowledge of a system, often represented by a simulation. In a diagnostic system, possible causes for a problem can be identified by reasoning backward from the symptoms to possible causes, and then simulating these causes to see if they produce the symptoms observed.
 - a. Such an approach can be especially helpful when it is necessary to identify the cause of a totally new problem, since there will not, of necessity, be heuristic rules or cases that apply.

Example: This often arises in debugging student programs.
 - b. However, such an approach can actually be less effective than a rule-based approach when diagnosing a problem, because of the myriad of detail involved.

Example: Many years ago, I worked on a custom system that was being installed at the Naval Air Development Center in Johnstown, PA. We were able to identify the problem as being due to a hardware anomaly.

I, being trained as an Electrical Engineer, tried to think about the problem from first principles, and - frankly - got nowhere. Then the technician who was with me said "Wait a minute - this looks like __" - and immediately solved the problem!

3. The book also mentioned the idea of a hybrid approach, using two or more of rule-based, case-based, and model-based reasoning. As the book noted, though, doing this is non-trivial!