CPS331 Lecture: Expert Systems last revised February 9, 2016

Objectives:

1. To introduce expert systems

Materials:

1. Projectable of Cawsey Figure 3.1
2. Projectable of animal identifier rules
3. Demo of Prolog identifier implementation
4. Projectable of bagger rules and test order
5. Demo of Prolog bagger implementation with test order
6. Projectable of MYCIN dialog from Winston (p. 193 f)
7. Projectable of And-Or tree version of a portion of the animal identifier rules
8. Demo of natural language isa hierarchy
9. Demo Prolog ES shell with identifier rules

I. Introduction

A. Today's topic is a bit different from previous topics in the course. Thus far, we have been talking about AI principles in the abstract. Today, we talk about a specific area of Applied AI: expert systems.

B. 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.

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.

C. We will consider three questions

1. What is an expert system?

2. Where are expert systems useful?

3. How is an expert system organized?

D. We will now look at a couple of _very simple_ examples. Both are based on examples in an AI text by Patrick Henry Winston (former director of the AI lab at MIT), though the Prolog implementation is mine. We first looked at both when we talked about rule-based systems - we will look at them again now, because expert systems are basically applied rule-based systems.

1. Identifier

   a) PROJECT Rules again

   b) DEMO Prolog implementation - including use of why and explanation of results.

      (1) tony (tiger)

      (2) polly (penguin)

2. Bagger.

   a) Background - Winston introduced this system this way.

      "Suppose we want Robbie, our robot, to bag groceries in the manner of a grocery-store checkout clerk. We are not interested in optimal packing, but we do want Robbie to know some of the fundamentals of grocery bagging: big bottles of Pepsi go in the bottom, with not too many in one bag; ice cream is protected in freezer bags; and little things are stuffed here and there when everything else is in place."

   b) PROJECT: Rules again

   c) PROJECT: Test order

   d) DEMO: Prolog implementation with test data set (goal = main.)

II. 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 are seeing, the field of AI is really much broader than that, and there have been numerous other areas of application of AI techniques.

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 condition-action rules of the sort we discussed earlier. That is, they are based on rules having the following general 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) Many expert systems allow their users to inspect their rules or provide some sort of explanation facility in order to allow the user 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.

   c) Contrast this with traditional software packages.

(1) 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.

The user of such a package must simply trust that the authors did the right thing (which, alas, is not always the case!)

(2) 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 one writer points 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 discuss 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), and as the title of a chapter in Cawsey, we will use it too.

D. Historically, expert systems stand at the opposite extreme, methodologically, from some of the earliest AI approaches

1. One of the earliest AI systems - created in 1957 - was called GPS or General problem solver. It was 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 a later writer put it, “not a modern control structure” (Winston). 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.)

3. The term "expert system" was once more prominent in the AI literature than it is today - perhaps because expert systems have become so integrated into software applications that they don't attract as much attention to themselves as they once did. (We might regard this as an example of a case where something that was once thought to require human expertise fades from view as being AI.)

III. 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.

   Example: Bagger is 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.

   Example: Identifier is such a system

a) An early example of such a system that is quite important historically was MYCIN.

   (1) 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.

   (2) 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.

   (3) 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.

   (4) 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."

   PROJECT - Sample Dialog

   (5) 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. One writer gives seven characteristics of a problem that is suitable for solution by an expert system. (Luger 6e page 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 traveling.

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.

C. 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 not 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:

      (1) Comparison of system output with correct answers given by the expert.

      (2) 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. A year to 18 months is fairly typical for the initial development of a system.

IV. How is an Expert System Organized?

A. Figure 3.1 in Cawsey shows a general paradigm for this type of software system:

   PROJECT

   1. Several parts of the system are relatively generic. Indeed, there are a number of "expert system shells" that contain implementations of these portions of the system. (This is what the dotted line in the diagram signifies.)

      a) The user interface handles interaction with the user. Typically, the program begins by asking the user various questions, and then reports its conclusions.

      b) The explanation system generates responses to two sorts of questions the user might ask.

(1) Why are you asking me this? (E.g. in a medical system, answering a question may require costly or unpleasant tests; therefore, it is important for the doctor using the system to know why the information is needed.)

(2) How did you reach this conclusion?

c) The inference engine is the heart of the system. It is relatively simple - though there are a number of different "styles" to choose from depending on the nature of the particular system - the most basic distinction being between forward-chaining and backward-chaining systems.

(1) Synthetic systems generally use a forward-chaining system, because the purpose is to develop a complete "plan" for a given set of requirements.

(2) Analytic systems may use backward-chaining systems, since the goal is to find the hypothesis that best fits a set of data; though forward-chaining systems are also used in many cases.

d) The system may or may not have a knowledge base editor, depending on the type of user the system is intended for.

2. The knowledge base contains the actual rules. This is the part of the system that is specific to a particular application, and is the main focus of the knowledge engineer's work (though there is usually also some need for customization of the shell)

3. The case-specific data pertains to one particular use of the system - e.g. in a medical diagnosis system, this part would contain information about the patient's symptoms, medical test results, etc.

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.

B. As we have already noted, 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.

This might be captured by a rule of the form:

```
if      engine cranks slowly
then    check battery condition
```

1. Notice that a heuristic like this is not infallible.

a) E.g., 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.)

c) Human experts are not infallible problem solvers, either. (I remember a rather unpleasant afternoon I spent in a gas station when my car refused to start again after I gassed up, and the mechanic applied this heuristic unsuccessfully ... the problem was actually due to a dirty connection on my starter motor, which was only discovered after replacing the battery didn't solve the problem!)

2. 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 <u>problem solver</u> (the expert) does, rather than on the <u>problem</u> itself.

      (1) 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.

      (2) 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 carburetor :-)

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

   Example: A portion of the rules for the "animal identifier" example we looked at earlier

   PROJECT: Original rules

   PROJECT: And Or Tree

   1. Each rule, in itself, is an <u>and</u> node, since all antecedents must hold before the consequent can be applied. This is symbolized by the arc connecting the antecedents. (In the case of rules having only one antecedent, there is no arc, of course, but the node is still considered an and node)

   2. When multiple rules lead to the same consequent, then the consequent is an <u>or</u> node: the consequent is true if any of the rules establishing it fire.

   3. And-or trees facilitate answering questions about the program's reasoning in the explanation system.

      a) "Why do you want to know" questions can be answered by giving the consequent of the rule that immediately gave rise to the question. The answer might take the form "because I am exploring the hypothesis that <consequent>." If the user asks "why" again, further answers can be generated by going up the tree toward the original goal.

         Example: if the animal identifier system is exploring the hypothesis that the animal is a cheetah and ends up asking the user "does the animal have hair", and the user asks "why do you want to know?", the explanation system can answer "Because I am exploring the hypothesis that the animal is mammal". If the user asks "why?" again, the response can be "because I am exploring the hypothesis that the animal is a carnivore". If the user again asks "why?", the answer would be "because I am exploring the hypothesis that the animal is a cheetah" A final "why" question could give rise to the answer "because you asked me to identify this animal".

b) "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.) If the user asks "how" again, further answers can be generated by going down the tree.

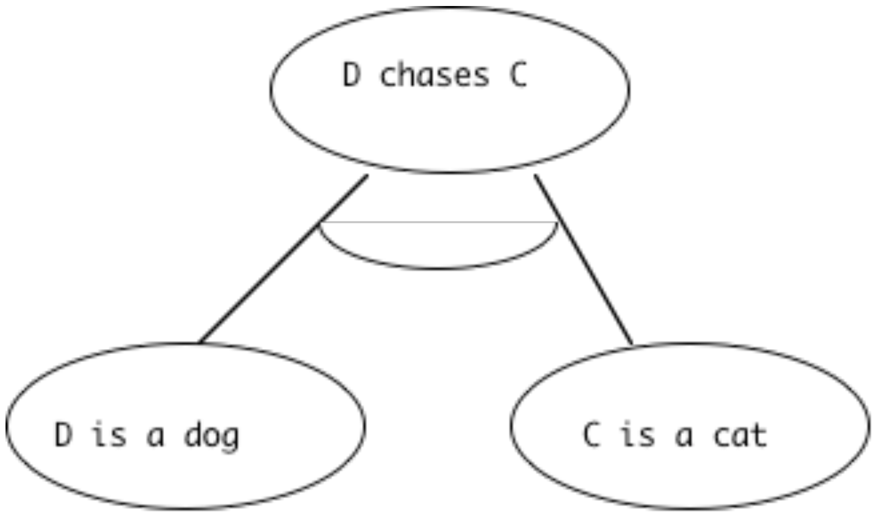
   Example: If the program reports that the animal is a cheetah, and the user asks "how did you reach this conclusion", the explanation system can report "because the animal is a carnivore and has a tawny color and dark spots". If the user asks how how the program concluded the animal is a carnivore, the explanation system can report "because it is a mammal and has hair" (assuming that I1 is the rule that fired) If the user asks how the program concluded that the animal is a mammal, the explanation system can answer "because it has hair". If the user asks a question like "how did you conclude the animal has dark spots?", the explanation system can say "Because you told me so".

4. This is the approach that lay behind the "explanation facility" in the natural language / isa hierarchy demo we have looked at previously.

   a) Run program - demo entry of several facts

      butch is a dog
      sylvester is a cat
      a dog chases cats

b) In effect, the rules in the database represent the following and/or tree



c) When we ask who chases sylvester, we get the answer butch. If we then ask why?, we get answers corresponding to the two children of the "chases" node.

DEMO

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 - e.g.the rules for the animal identifier system include some initial rules to determine the general category of the animal (mammal, bird ...), followed by specific rules for different kinds of animal (tiger, cheetah ...)

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:

(1) 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.

Example: If I5 fails because it is not known that the animal eats meat, I6 can still be tried.

(2) Reject all rules in the group - if the failure was caused by an antecedent that is common to all members of the group.

Example: If I5 fails because it cannot be shown that the animal is a mammal, then I6 need not even be tried.

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. (Recall that the Bagger system uses this approach with its notion of "step").

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.

   (1) For synthetic systems, this is certainly appropriate.

   (2) For analytic systems, this may result in an undesired explosion of information.

   (3) The Bagger demo used forward chaining

2. Backward chaining systems are often useful for analysis.

   a) Such systems may get information from the user by asking specific questions, as Identifier 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.

      (1) 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.)

      (2) 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.]

Example: Suppose the animal identifier system asked the user "does the animal have hair?" and "does the animal give milk?" while pursuing the hypothesis that the animal is a cheetah. If the user answered "no" to both, there would be no need to ask again while pursuing the hypothesis that the animal is a cheetah; rather the program could abandon that hypothesis at once.

(3) The search continues until one hypothesis is shown to be true or until all hypotheses are rejected.

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 the basic control regimen of Prolog. This makes Prolog a nice tool for building such systems. However, as was the case with Bagger, Prolog can also be used for forward-chaining systems, with a bit more work.

V. **Expert System Shells**

A. As the diagram of expert system structure in Cawsey indicated, expert systems often make use of a "shell" which is problem-independent plus a set of problem-specific rules.

PROJECT Cawsey figure 3.1 again

B. There are, in fact, a large number of commercial and freeware expert system shells available.

DEMO Google search on "expert system shell"

1. Some are quite sophisticated (and expensive)

2. Most require some amount of expertise on the part of the user for formulating rules.

3. For demonstration purposes (and a homework problem and project), we will be using a very simple shell that works with rules written in Prolog.

C. DEMO: ES Shell

1. Show answering a query (answer according to tony - a tiger)

   a) Demo "Why" for a question

   b) Demo "Why" for a conclusion

2. Show structure of rules

3. Show what happens if the rules don't cover a case at all (roxie)

4. Show what happens if the rules are incomplete (tweety - a canary inferred to be an albatross).

5. Modify rules to recognize a dog (add dog if carnivore and barks) - demonstrate

VI. **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

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-accessible 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.

    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: While in college, I worked on a custom system that was being installed at the Naval Air Development Center in Johnstown, PA. We ran into a problem that we were able to identify 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. It is also possible to use a hybrid approach, using two or more of rule-based, case-based, and model-based reasoning. Doing this, though, is non-trivial!