Expert Systems

• General Characteristics
  – As the name implies, they try to provide solutions to problems of the same quality as experts in the problem domain
  – It should be possible to inspect how an expert system reached a conclusion and this should be presented in a logically plausible sequence of reasons
  – It should be easy to modify the reasoning strategies used by the expert system, this means we need to separate knowledge from control
  – reasoning strategies should be based on heuristic knowledge, not just pure logic
  – we won’t attempt to model the insight often possessed by an expert that allows her to jump directly to some conclusions

Problem Domains

• Expert systems have been applied to medical diagnosis (MYCIN), chemical analysis (DENDRAL), oil drilling (Prospector), computer configurations (XCON), and many other areas

1. **Interpretation**—forming high-level conclusions or descriptions from collections of raw data.
2. **Prediction**—projecting probable consequences of given situations.
3. **Diagnosis**—determining the cause of malfunctions in complex situations based on observable symptoms.
4. **Design**—finding a configuration of system components that meets performance goals while satisfying a set of design constraints.
5. **Planning**—devising a sequence of actions that will achieve a set of goals given certain starting conditions and run-time constraints.
6. **Monitoring**—comparing a system’s observed behavior to its expected behavior.
7. **Debugging and Repair**—prescribing and implementing remedies for malfunctions.
8. **Instruction**—detecting and correcting deficiencies in students’ understanding of a subject domain.
9. **Control**—governing the behavior of a complex environment.
Expert System Architecture

- Design goals
  - use natural reasoning strategies such as if..then, “if the light doesn’t come on, check the bulb”
  - capturing the knowledge base can be separate from implementing the inference engine
  - the same inference engine can be applied to a variety of problem domains
  - modularity allows us to test different control strategies to obtain the best results

Select Appropriate Problems

- Their must be sufficient need - expert systems aren’t toys, a good system is expensive to develop and must pay off
- Human expertise is not available - use the expert system as a substitute
- Problems are appropriate for symbolic computation - minimize the need for perception or dexterity
- The problem domain is susceptible to logical reasoning
- The problem goes beyond a simple computational problem
- Experts to help build the system are available and willing participants
- The problem is of appropriate size and scope
Knowledge Engineering
• Putting expertise into an expert system is not a one step process; rather than are many feedback loops and correction steps
  • The first step is to get a working prototype that appears to be headed in the right direction
  • Even after the prototype is acceptable, there are many steps of refinement before the final evaluation

A Naïve View
• At first glance it may seem that an expert system is just a set of if..then rules that have to be put into the system and refined

• In practice, such an approach will lead to tweaking the system back and forth without getting acceptable performance
• What is lacking in this naïve approach is a conceptual model of the problem domain
Using a Conceptual Model

• A conceptual model is not executable, rather it is a design construct

• As a design document, the conceptual model should be publicized

• This model should be reflected in other components, such as data dictionaries, search space representations, and comments in the code regarding implementation

Approaches to Expert Systems

• Rule-based systems
  – knowledge is in the form of if..then rules
  – users can be asked to supply data
  – users can ask why and how
  – search can be goal or data directed
  – various control heuristics are possible

• Model-based systems
  – Model based on conceptual components
  – Use input/output behavior to detect underlying conceptual model
  – Models may include both correct and incorrect reasoning patterns

• Case-based systems
  – Uses previously encountered situations
  – An appropriate case may be retrieved and modified to fit current circumstances
  – Successful modified cases may be saved
Group Work

• Suppose you want to build an expert system to determine a type of plant
• What conceptual model would you use?

Example of a Goal-driven system

• In this simplified domain of diagnosing automotive problems then are only four rules, as shown below

  Rule 1: if
  the engine is getting gas, and
  the engine will turn over,
  then
  the problem is spark plugs.

  Rule 2: if
  the engine does not turn over, and
  the lights do not come on
  then
  the problem is battery or cables.

  Rule 3: if
  the engine does not turn over, and
  the lights do come on
  then
  the problem is the starter motor.

  Rule 4: if
  there is gas in the fuel tank, and
  there is gas in the carburetor
  then
  the engine is getting gas.
Group Work

• Give some if .. then rules that would be appropriate for an expert system to identify the type of a plant

Starting the Production System

• The goal, to find the problem, is added to memory with the unbound variable X
• Rules 1,2,3 can fire, we select 1
• This adds subgoals the engine is turning over and the engine is getting gas
• If rule 1 ultimately fails, we would have to backtrack and try rule 2
Querying the User

- The engine getting gas causes rule 4 to fire adding subgoals is there fuel in the tank and gas in the carburetor
- These facts are not known, so the system can query the user
- The reasoning for this system is easily expressed in an and/or graph

More User Interaction

- In addition to asking the user questions, the user can ask the system about its reasoning
- If the user asks “why” to a query, the system should provide the chain of reasoning
- The user can also ask the chain of reasoning for a conclusions, such as how the engine is getting gas
- This dynamic behavior is easy to implement in Lisp or Prolog, as we will see later on
Group Work

- In determining a plant type, what questions would you want to ask the user?

Data-driven reasoning

- The working memory is initially empty
- Any part of a premise is “askable” if it is not the conclusion of another rule
- The first premise in rule 1 is not askable since it is the conclusion for rule 4
- The first premise of rule 2 is evaluated by asking the user
- Suppose the user responds the engine does turn over
- Rule 3 doesn’t fire so we move to Rule 4 which has two askable premises
- Suppose the responses from the user are both true
Opportunistic Reasoning

- This search strategy is based on the following rule: when new information is concluded control moves to those rules that have the information as a premise
- Thus the concluded information controls the direction of the search
- Unlike pure data driven searches, this search is focused on reaching a goal

Use of Heuristics

- Ordering of premises: if p and q then r is executed by trying p first, so often the most restrictive premise will be put first to narrow the search
- Ordering of rules: our automotive example had poor rule ordering since it makes more sense to find out whether the engine is turning over before pursuing “getting gas”
- A four stage approach:
  - organize situation, collect the data, perform the analysis, report the conclusions
  - by adding control premises to the rules, an ordinary inference engine will work
- Other heuristics: refraction, recency, and specificity
Human Expertise

• Theoretical knowledge
• Experience-based problem-solving heuristics
• Past problems and their solutions
• Perceptual and interpretive skills
• Intuition
• What else?

Model-based Reasoning

• Limitations of rule-based systems
  – if the problem instance was not anticipated, the system failed
  – heuristics were applied inappropriately
  – a deeper, more theoretical understanding of the problem domain is not used
• Model-based reasoning attempts to overcome these limitations
• Early efforts involved intelligent tutoring systems (ITS)
• Model-based analysis requirements
  – a description of each component
  – a description of the internal structure
  – diagnosing a particular problem is based on input/output behavior
Example-photo enlargements

• In a photograph a father is 8 cm high and his daughter is 6 cm high.
• In an enlargement, the father is 12 cm high. How high is the daughter?
• What is the underlying model given the following answers:
  – 10 cm, the father grew by 4 cm so the daughter grows by 4 cm.
  – 9 cm, the picture is 1.5 times larger since 8 \times 1.5 = 12 cm, so the daughter is 6 \times 1.5 = 9 cm.
  – 10 cm, the father is 2 cm higher than the daughter so 12 - 2 = 10 cm.
  – 9 cm, the daughter is 3/4 the height of the father, so 12 \times 3/4 = 9 cm.

Behavior of an Adder

• Any of the following characterizations captures the correct behavior of an adder:
  If we know the values at A and B, the value of C is A + B (the solid line).
  If we know C and A the value at B is C – A (the dashed line).
  If we know C and B, the value at A is C – B (the dotted line).
A more complex circuit

- Inputs are A through E, the outputs are F and G
- There are three multipliers followed by two adders
- We expect 12 at F, but the result is 10; what is wrong?
  - One of the devices, add-1, mult-1 or mult-2 must be faulty
  - Based on the correct output at G, we conclude that mult-2 is not at fault
  - We need to collect more data with different values to determine if mult-1 or add-1 is at fault
  - Give a set of test data to make this discrimination

Determining the cause of faults

1. Hypothesis generation, in which, given a discrepancy, we hypothesized which components of the device could have caused it.
2. Hypothesis testing, in which, given a collection of potential faulty components, we determined which of them could have explained the observed behavior.
3. Hypothesis discrimination, in which when more than one hypothesis survives the testing phase, as happened in the case of Figure 6.14, we must determine what additional information can be gathered to continue the search for the fault.

- The problem becomes much more complex if multiple faults are involved
- Systems based on models are more robust than systems based on heuristics
- However, model-based reasoning requires more data collection and is less efficient
- Model-based reasoning is limited to the model itself and cannot account for other errors such as “bridging faults”
Case-based Reasoning

- Early work from Roger Shank’s group at Yale on scripts and mops
  - the restaurant script
  - why mops (memory organizational packets) are needed
- Medicine and law are two disciplines that rely heavily on previous cases
- Case-based systems can learn from experience
- Solution steps
  - retrieve appropriate cases from memory
  - modify the case to match current circumstances
  - apply transformed case to new problem
  - save solution as well as success or failure

Organizing case data

- Possible strategies
  1. Goal-directed preference. Organize cases, at least in part, by goal description. Retrieve cases that have the same goal as the current situation.
  2. Salient-feature preference. Prefer cases that match the most important features or those matching the largest number of important features.
  3. Specify preference. Look for as exact as possible matches of features before considering more general matches.
  4. Frequency preference. Check first the most frequently matched cases.
  5. Recency preference. Prefer cases used most recently.
  6. Ease of adaptation preference. Use first cases most easily adapted to the current situation.
- Continued knowledge acquisition is relatively easy
- Some difficulties with CBR are
  - determining relevant features
  - efficiencies in the cost of storage versus the savings in computation
  - explanations are fairly weak
Transformational analogy

- Similar to learning by analogy
- In the work by Carbonell, the goal is to transform a source solution into a solution for the target problem

- Operators modify solutions by inserting or deleting steps in the solution path

Advantages of Rule-based Reasoning

1. The ability to use, in a very direct fashion, experiential knowledge acquired from human experts. This is particularly important in domains that have not been well formalized, or that rely heavily on heuristics to manage complexity or missing information.

2. The modularity of rules eases construction and maintenance. This supports the iterative development cycle required for expert systems.

3. Good performance is possible in limited domains. Because of the large amounts of knowledge required for intelligent problem-solving, expert systems are limited to narrow domains. However, there are many such domains where design of an appropriate system can prove extremely useful.

4. Good explanation facilities. Although the basic rule-based reasoning framework supports flexible, problem-specific explanations, it must be mentioned that the ultimate quality of these explanations depends upon the structure and content of the rules. Knowledge engineers must pay attention to these issues in designing rules. We have also shown how explanation facilities differ widely between data-driven and goal-driven rule systems.

5. Rules map naturally into state space search.

6. Rule chaining is fairly easy to trace and debug. Good explanation facilities further support debugging.

7. Steps within the problem solution process are open to inspection. This supports both explanations and debugging. It also gives user interface designers a very flexible means of designing interactive user dialogs.

8. The separation of knowledge from control further simplifies development of expert systems by enabling an iterative development process where the engineer acquires, implements and tests individual rules.
Disadvantages of Rule-based Reasoning

1. Often the rules obtained from human experts are highly heuristic in nature, and lack a deeper, functional knowledge of the domain.
2. Heuristic rules tend to be "brittle" and cannot handle missing information or unexpected data values.
3. Another aspect of the brittleness of rules is a tendency to degrade rapidly near "edges" of the domain knowledge. Unlike humans, rule-based expert systems are usually unable to fall back on first principles when confronted with novel problems.
4. Formalization function at the descriptive level only, omitting deeper, theoretical
5. The knowledge tends to be very task dependent. Unlike human intelligence, formalized domain knowledge tends to be very specific in its applicability. Currently, knowledge representation languages cannot approach human flexibility.

Case-based Reasoning

• Advantages of case-based reasoning

1. The ability to encode historical knowledge directly. In many domains, cases can be obtained from existing case histories, repair logs, or other sources, eliminating the need for intensive knowledge acquisition with a human expert.
2. Allows shortcuts in reasoning. If an appropriate case can be found, new problems can often be solved in much less time than it would take to generate a solution from rules or models.
3. It allows a system to avoid past errors and exploit past successes. CBR provides a model of learning that is both theoretically interesting and practical enough to apply to complex problems.
4. Extensive analysis of domain knowledge is not required. Unlike a rule-based system, where the knowledge engineer must anticipate rule interactions, CBR allows a simple accumulative model of knowledge acquisition. This happens once an appropriate representation for cases, a useful retrieval index, and a case adaptation strategy are designed.
5. Knowledge acquisition and coding are relatively easy.
6. Appropriate indexing strategies add insight and problem-solving power. The ability to distinguish differences in target problems and select an appropriate case is an important source of a case-based reasoner’s power; often, indexing algorithms can provide this functionality automatically.

• Disadvantages of case-based reasoning

1. Cases lack deeper knowledge of the domain. This handicaps explanation facilities. In many situations, it allows the possibility that cases may be misapplied, leading to wrong or poor quality advice.
2. A large case base can suffer problems from store/compute trade-offs.
3. It is difficult to get good criteria for indexing and matching of cases. Currently, retrieval vocabularies and similarity matching algorithms must be carefully hand crafted. This can offset many of the advantages CBR offers for knowledge acquisition.
Model-based Reasoning

- Advantages of model-based reasoning

1. The ability to use functional/structural knowledge of the domain in problem-solving. This increases the reasoner's ability to handle a variety of problems, including those that may not have been anticipated by the system's designers.

2. Unlike rule-based expert systems, model-based reasoners tend to be very robust. For the same reasons that humans often retreat to first principles when confronted with a novel problem, model-based reasoners tend to be robust and flexible problem solvers.

3. Some knowledge is transferable between tasks. Model-based reasoners are often built using scientific, theoretical knowledge. Because science strives for generally applicable theories, this generality often extends to model-based reasoners.

4. Often, model-based reasoners can provide causal explanations. These can convey a deeper understanding of the fault to human users, and can also play an important tutorial role (see also Section 16.2).

- Disadvantages of model-based reasoning

1. A lack of experiential (descriptive) knowledge of domain. The heuristic methods used by rule-based approaches reflect a valuable class of expertise.

2. It requires an explicit domain model. Many domains, such as the diagnosis of failures in electronic circuits, have a strong scientific basis that supports model-based approaches. However, many domains, such as some medical specialties, most design problems or many financial applications lack a well defined scientific theory. Model-based approaches cannot be used in such cases.

3. High complexity. Model-based reasoning generally operates at a level of detail that leads to significant complexity; this is, after all, one of the main reasons human experts develop heuristics in the first place.

4. Exceptional situations. Unusual circumstances, such as a bridging fault in an electronic circuit, can alter the functionality of a system in ways impossible to predict a priori.

Hybrid Reasoning

- Rule-based and case-based systems

1. Offer a natural first check against known cases before undertaking rule-based reasoning and the associated search costs.

2. Provide a record of examples and exceptions to solutions through the case base.

3. Assist in addressing incomplete or inconsistent reasoning situations by resorting to cases that have proven useful in the past.

4. Record search-based results as cases for future use. By saving appropriate cases, a reasoner can avoid duplicating costly search.

- Rule-based and model-based systems

1. Enhance explanations with functional knowledge. This can be particularly useful in tutorial applications.

2. Improve robustness when rules fail. If there are no heuristic rules that apply to a given problem instance, the reasoner can resort to reasoning from first principles.

3. Add heuristic search to model-based search. This can help manage the complexity of model-based reasoning and allow the reasoner to choose intelligently between possible alternatives.

- Model-based and case-based systems

1. Give more mature explanations to the situations recorded in cases.

2. Offer a natural first check against stored cases before beginning the more extensive search required by model-based reasoning.

3. Provide a record of examples and exceptions in a case base that can be used to guide model-based inference.

4. Record results of model-based inference for future use.