Expert Systems

Byoung-Tak Zhang

TA: Hyo-Sun Chun

School of Computer Science and Engineering
Seoul National University
Expert Systems

• Computer software that:
  – Emulates human expert
  – Deals with small, well defined domains of expertise
  – Is able to solve real-world problems
  – Is able to act as a cost-effective consultant
  – Can explains reasoning behind any solutions it finds
  – Should be able to learn from experience.
Expert Systems

• A computer system that emulates the decision-making ability of a human expert
  – Designed to solve complex problems by reasoning about knowledge, represented primarily as if–then rules rather than through conventional procedural code

• An example of a knowledge-based system
  – The first commercial systems to use a knowledge-based architecture

A Symbolics Lisp Machine: An Early Platform for Expert Systems. Note the unusual "space cadet keyboard".
• Divided into two sub-systems:
  – Knowledge base
    • The knowledge base represents facts and rules.
  – Inference engine
    • The inference engine applies the rules to the known facts to deduce new facts.
    • Inference engines can also include explanation and debugging capabilities.
Components of an Expert System

Expert System

Knowledge Base

Inference Engine

User Interface

User
Knowledge Base

• A technology used to store information used by a computer system
• Represents facts about the world
• In early expert systems, flat assertions about variables
• In later expert systems, more structure and utilized concepts from object-oriented programming
Inference Engine

• An automated reasoning system
  – Evaluates the current state of the knowledge-base
  – applies relevant rules
  – asserts new knowledge into the knowledge base

• Capabilities for explanation
  – can explain to a user the chain of reasoning
  – forward chaining: data-driven
  – backward chaining: goal-driven

[Wikipedia]
Various types of inference engines

• **Truth Maintenance**
  – Truth maintenance systems record the dependencies in a knowledge-base so that when facts are altered dependent knowledge can be altered accordingly.

• **Hypothetical Reasoning**
  – In hypothetical reasoning, the knowledge base can be divided up into many possible views.
  – This allows the inference engine to explore multiple possibilities in parallel.

• **Fuzzy Logic**
  – One of the first extensions of simply using rules to represent knowledge was also to associate a probability with each rule.

• **Ontology Classification**
  – With the addition of object classes to the knowledge base a new type of reasoning was possible.
  – Rather than reason simply about the values of the objects the system could also reason about the structure of the objects as well.
Problem Domain vs. Knowledge Domain

• An expert’s knowledge is specific to one problem domain – medicine, finance, science, engineering, etc.

• The expert’s knowledge about solving specific problems is called the knowledge domain.

• The problem domain is always a superset of the knowledge domain.
Knowledge Engineering

- The process of building an expert system:
  - The knowledge engineer establishes a dialog with the human expert to elicit (obtain) knowledge.
  - The knowledge engineer codes the knowledge explicitly in the knowledge base.
  - The expert evaluates the expert system and gives a critique to the knowledge engineer.
Development of an Expert System
The Role of AI

• An algorithm is an ideal solution guaranteed to yield a solution in a finite amount of time.
• When an algorithm is not available or is insufficient, we rely on artificial intelligence.
• Expert system relies on inference (conclusion) – we accept a “reasonable solution.”
Uncertainty

• Both human experts and expert systems must be able to deal with uncertainty.
• It is easier to program expert systems with shallow knowledge than with deep knowledge.
• Shallow knowledge – based on empirical and heuristic knowledge.
• Deep knowledge – based on basic structure, function, and behavior of objects.
Advantages of Expert Systems

• Increased availability
• Reduced cost
• Reduced danger
• Performance
• Multiple expertise
• Increased reliability
• Explanation
• Fast response
• Steady, unemotional, and complete responses at all times
• Intelligent database
Problems with Expert System

• Limited domain
• Systems are not always up to date, and don’t learn
• No “common sense”
• Experts needed to setup and maintain system
• Who is responsible if the advice is wrong?
## Applications

<table>
<thead>
<tr>
<th>Category</th>
<th>Problem Addressed</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpretation</td>
<td>Inferring situation descriptions from sensor data</td>
<td>Hearsay (Speech Recognition), PROSPECTOR</td>
</tr>
<tr>
<td>Prediction</td>
<td>Inferring likely consequences of given situations</td>
<td>Preterm Birth Risk Assessment</td>
</tr>
<tr>
<td>Diagnosis</td>
<td>Inferring system malfunctions from observables</td>
<td>CADUCEUS, MYCIN, PUFF, Mistral, Eydenet, Kaleidos</td>
</tr>
<tr>
<td>Design</td>
<td>Configuring objects under constraints</td>
<td>Dendral, Mortgage Loan Advisor, R1 (Dec Vax Configuration)</td>
</tr>
<tr>
<td>Planning</td>
<td>Designing actions</td>
<td>Mission Planning for Autonomous Underwater Vehicle</td>
</tr>
<tr>
<td>Monitoring</td>
<td>Comparing observations to plan vulnerabilities</td>
<td>REACTOR</td>
</tr>
<tr>
<td>Debugging</td>
<td>Providing incremental solutions for complex problems</td>
<td>SAINT, MATHLAB, MACSYMA</td>
</tr>
<tr>
<td>Repair</td>
<td>Executing a plan to administer a prescribed remedy</td>
<td>Toxic Spill Crisis Management</td>
</tr>
<tr>
<td>Instruction</td>
<td>Diagnosing, assessing, and repairing student behavior</td>
<td>SMH.PAL, Intelligent Clinical Training, STEAMER</td>
</tr>
<tr>
<td>Control</td>
<td>Interpreting, predicting, repairing, and monitoring system behaviors</td>
<td>Real Time Process Control, Space Shuttle Mission Control</td>
</tr>
</tbody>
</table>
Figure 16-2 An expert system on the job. This expert system helps Ford mechanics track down and fix engine problems.

Figure 16-3 Airline scheduling program produced with the aid of an expert system. This system offers a graphical user interface to help solve a complex airport scheduling problem. (a) This screen illustrates the system's ability to display multiple views of objects and the relationships between them. (b) Various screen windows show planes circling the airport, the number of planes approaching the airport, gate information, and two concourses with planes at their gates.
Expert System Tools

- Algorithmic languages
  - (such as 'C', Pascal, Basic)
- Symbolic languages
  - (such as Prolog, LISP)

- Development Environments
  - (such as Art, KEE, LOOPS)
- Expert System Shells
  - (such as Crystal, XpertRule, Leonardo, Xi-Plus)
Algorithmic Languages

• Flexible and powerful
• They can be used to tailor a system exactly to an application
• Lacking in knowledge engineering framework.
Symbolic Languages

• Computer languages for logic programming must have structures for storing and retrieving known and deduced facts from a fact base or knowledge base, and they must have functions or procedures for deducing new facts.

• LISP
• Prolog
Development Environments

• Expert system programming environments are special packages of pre-written code. They are "power tools" for building knowledge-based systems.
• They provide a set of building blocks that cater for the programmers needs, and hence are known as ‘tool kits’.
• In most cases, the price to pay is the loss of flexibility, however, in using a higher-level tool.
Expert System Shells

- Shells are tools for building expert systems that provide knowledge representation facilities and inferencing mechanisms.
- The programmer must gain detailed knowledge about a particular domain from an expert and information source.
- May be thought of as an expert system with all the domain specific knowledge removed and a facility for entering a new knowledge-base provided.
History of Expert Systems

1. Early to Mid-1960s
   - One attempt: the General-purpose Problem Solver (GPS)
   - **General-purpose Problem Solver (GPS)**
   - A procedure developed by Newell and Simon [1973] from their Logic Theory Machine -
     - Attempted to create an "intelligent" computer
       - general problem-solving methods applicable across domains
     - Predecessor to ES
     - Not successful, but a good start
2. Mid-1960s: Special-purpose ES programs
   - DENDRAL
   - MYCIN

Researchers recognized that the problem-solving mechanism is only a small part of a complete, intelligent computer system
   - General problem solvers cannot be used to build high performance ES
   - Human problem solvers are good only if they operate in a very narrow domain
   - Expert systems must be constantly updated with new information
   - The complexity of problems requires a considerable amount of knowledge about the problem area
3. *Mid 1970s*

- Several Real Expert Systems Emerge
- Recognition of the Central Role of Knowledge
- AI Scientists Develop
  - Comprehensive knowledge representation theories
  - General-purpose, decision-making procedures and inferences

**Limited Success Because**
- Knowledge is Too Broad and Diverse
- Efforts to Solve Fairly General Knowledge-Based Problems were Premature
BUT

- Several *knowledge representations* worked

**Key Insight**

- *The power of an ES is derived from the specific knowledge it possesses, not from the particular formalisms and inference schemes it employs*
4. Early 1980s

- **ES Technology Starts to go Commercial**
  - XCON
  - XSEL
  - CATS-1

- **Programming Tools and Shells Appear**
  - EMYCIN
  - EXPERT
  - META-DENDRAL
  - EURISKO

- **About 1/3 of These Systems Are Very Successful and Are Still in Use**
Latest ES Developments

- Many tools to expedite the construction of ES at a reduced cost
- Dissemination of ES in thousands of organizations
- Extensive integration of ES with other CBIS
- Increased use of expert systems in many tasks
- Use of ES technology to expedite IS construction (*ES Shell*)

[Decision Support Systems and Intelligent Systems, E. Turban and J. E. Aronson]