• Joint Probability Distribution (JPD)
• Full Joint Probability Distribution Table “FJPDT”
• Reasoning using the “FJPDT”
• Absolute and Conditional Independence
• Limitations of the “FJPDT” Technique
• Bayes’ Rule
• Causal and Diagnostic Knowledge
• Reasoning using General Bayes’ Rule
• Naïve Bayes Classifier
• Bayesian Networks Representation and Construction
• Algorithm for Constructing: Bayesian Networks
• Knowledge Engineering for Building Belief Networks
• Computing Joint Probabilities: Using a Bayesian Network
• Probabilistic Reasoning using Bayesian Networks
  • Causal (Top-Down) Inference
  • Diagnostic (Bottom-Up) Inference
• Independence in a Bayesian Network
• Tradeoff of FJPDT vs. Bayesian Network
Overview of Expert Systems

- Introduction to Expert Systems (ES)
- Criteria for Building an Expert System
- Why and When to Use an Expert System?
- Architecture of an Expert System
- Components of an ES
- Inference Engine Control Strategies
- Building an ES
- MYSIN as a Case Study
Introduction: ES Definition

• Definitions

• A computer program designed to model the problem-solving ability of a human expert.

  *Expert Systems Design and Development by Durkin*

• A model and associated procedure that exhibits, within a specific domain, a degree of expertise in problem solving that is comparable to that of a human expert.

  *Expert Systems by Ignizio*

• A computer system which emulates the decision-making ability of a human expert.

  *Expert Systems: Principles and Programming by Giarratano and Riley*
Criteria for Building an Expert System

- Does a human know how to solve the problem?
  - If no human expert exists, it is not possible to develop rules describing the problem.
  - The techniques of solving the problem must be known and defined in order to create an expert system.
- Does the problem have a definable solution?
  - If all of the possible solutions cannot be specified; writing rules to solve the problem is difficult.
- Is the level of understanding and scope appropriate?
  - A problem that has too wide scope or requires too deep level of understanding is not appropriate for an ES.
  - An ES solves specific problems.
- Has the technique for solving the problem been documented?
  - Solution may be a decision tree, manual procedure, written instructions, etc.
  - Well-defined problems can be easily converted to an ES.
• Problem should NOT have too large a scope or too deep a level of understanding.
• Defining the problem to fall within the shaded area of the graph is very important.
Why Use an Expert System?

- Frees expert from repetitive, routine jobs.
- Provides the beginner with expert advice on a specific subject.
- Wide distribution of rare human knowledge.
- Aids in training new employees.
- Improves worker productivity.
- Provides second opinion in critical situations.
  - Especially valuable when tired or under stress.
Technical Advantages of an Expert System

- Rapid prototype development
- Easier verification of software
- Easier maintenance of software
- Explains its reasoning in English to user when requested.
- Truly self documenting software.
- Easier to learn to build rule-based expert systems.
- Inexpensive technology.
- Automated consistency checking of knowledge in the KB.
## Comparison of a Human Expert and an Expert System

<table>
<thead>
<tr>
<th>Factor</th>
<th>Human Expert</th>
<th>Expert System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time availability</td>
<td>Workday</td>
<td>Always</td>
</tr>
<tr>
<td>Geographic</td>
<td>Local</td>
<td>Anywhere availability</td>
</tr>
<tr>
<td>Safety</td>
<td>Irreplaceable</td>
<td>Replaceable</td>
</tr>
<tr>
<td>Perishable/consumable</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Performance</td>
<td>Variable</td>
<td>Consistent</td>
</tr>
<tr>
<td>Speed</td>
<td>Variable</td>
<td>Consistent (usually faster)</td>
</tr>
<tr>
<td>Cost</td>
<td>High</td>
<td>Reasonable</td>
</tr>
</tbody>
</table>
Disadvantages of an ES

• Brittleness
  – Only have access to highly specific domain knowledge
  – Cannot fall back on more general knowledge when needed.

• Lack of Meta-Knowledge
  – Do not have sophisticated knowledge about their own operation.
  – Cannot reason about their own scope and limitations.

• Knowledge Acquisition is a bottleneck in applying ES technology to new domains

• Validation of an ES is difficult.

• Expert Systems can make mistakes.
When to use an Expert System?

- Final users agree that payoff will be high
- Application is knowledge intensive
- A human expert exists
- Not a natural-language intensive application.
- A wide range of test cases are available.
- Neither creativity nor physical skills are required.
Personnel Involved

- **Domain Expert**
  - A person who possesses some skills that allow him/her to draw upon past experiences and quickly focus on the core of a given problem.
  - Anyone can be considered a **domain expert** if he or she has deep knowledge (of both facts and rules) and strong practical experience in a particular domain.

- **User**
  - A person who will use the expert system and eventually benefit from the domain expert’s knowledge.

- **Knowledge Engineer**
  - A person who designs, develops, and implements expert systems (or other artificial intelligent applications).
The main players in the development team

Expert System Development Team

- Project Manager
- Domain Expert
- Knowledge Engineer
- Programmer

Expert System

End-user
The main players in the development team

1. **Knowledge engineer:**
   1. Capable of designing, building and testing an expert system.
   2. He or she interviews the domain expert to find out how a particular problem is solved.
   3. The knowledge engineer establishes what reasoning methods the expert uses to handle facts and rules and decides how to represent them in the expert system.
   4. The knowledge engineer then chooses some development software or an expert system shell, or looks at programming languages for encoding the knowledge.
   5. And finally, the knowledge engineer is responsible for testing, revising and integrating the expert system into the workplace.

2. **Programmer:**
   1. responsible for the actual programming, describing the domain knowledge in terms that a computer can understand.
The main players in the development team

1. The programmer needs to have skills in symbolic programming in such AI languages as LISP, Prolog and OPS5, some experience in the application of different types of expert system shells. In addition, the programmer should know conventional programming languages like C++, Java, and …etc.

1. **Project manager:** is the leader of the expert system development team,
   1. Responsible for keeping the project on track.
   2. He or she makes sure that all deliverables and milestones are met, interacts with the expert, knowledge engineer, programmer and end-user.

2. **End-user:**
   1. Person who uses the expert system when it is developed.
   2. The design of the user interface of the expert system is vital for the project’s success; the end-user’s contribution here can be crucial.
Phases in Expert System Development

1. Phase 1: Assessment
2. Phase 2: Knowledge Acquisition
3. Phase 3: Design
4. Phase 4: Test
5. Phase 5: Documentation
6. Phase 6: Maintenance

Requirements → Reformulations
Knowledge → Explorations
Structure → Refinements
Evaluation → Product
A Knowledge Engineer’s View of Intelligence

- Decisions
- Knowledge
- Facts
- "Raw" Data

+ Analysis/Deduction
+ Rules
+ Context
The End!!

Thank you

Any Questions?
ICS-381
Principles of Artificial Intelligent

Week 14.2

Overview of Expert Systems

A Part of Chapter 16: Making a Simple Decision

Dr. Tarek Helmy El-Basuny
Last Class

- Introduction to Expert Systems (ES)
- Criteria for Building an Expert System
- Why and When to Use an Expert System?
- The main players in the development team
- Phases in Expert System Development
Today’s Agenda

- Architecture of an Expert System
- Components of an Expert System
- Knowledge Representation
  - IF-THEN Rules as a Knowledge Representation
  - Case Base as a Knowledge Representation
- The Case Base Reasoning Cycle
- Base Inference Engine Control Strategies
  - Backward Chaining
  - Forward Changing
- Meta Rules: Rules about how to apply a strategy to avoid confliction resolution.
- Building an ES
  - Analysis
  - Specification
  - Development
  - Deployment
- MYSIN as a Case Study
22

Architecture of an Expert System

User Interface may employ:
- Question-and-Answer
- Menu-Driven
- Natural Language
- Graphics User Interface

User

Knowledge Engineer

Knowledge-Base

Inference Engine

Explanation Facilities

General Knowledge Base

Case-specific Data

DR. Tarek Helmy@KFUPM-ICS
Components of an Expert System

- **Knowledge base:**
  - Contains the domain knowledge useful for problem solving.
  - In a rule-based expert system, the knowledge is represented as a set of rules.
  - Each rule specifies a relation, recommendation, directive, strategy or heuristic and has the **IF (condition) THEN (action) structure**.
  - **ELSE** part is the same as the **THEN** part and is applied if any of the **IF** conditions are **FALSE**.
  - **ELSE** part is optional and not needed in most rules.
  - A rule can have multiple conditions joined by the keywords **AND (conjunction)**, **OR (disjunction)** or a combination of both.

```
IF <Condition 1>  IF <Condition1>
AND <Condition 2>  OR <Condition 2>
  ...
AND <Condition n>  OR <Condition n>
THEN <Action 1>    THEN <Action 1>
ELSE <Action 2>    ELSE <Action 2>
```
Components of an Expert System

- **Inference engine:**
  - Carries out the reasoning whereby the expert system reaches a solution.
  - Expert systems employ symbolic reasoning when solving a problem.
  - **Symbols** are used to represent different types of knowledge such as facts, concepts and rules.
  - It links the rules given in the knowledge base with the facts provided by the user.

- **Explanation facilities:**
  - An expert system must be able to explain its reasoning and justify its advice, analysis or conclusion.
  - Enable the user to ask the expert system how a particular conclusion is reached and why a specific fact is needed.

- **User interface:**
  - The means of communication between a user seeking a solution to the problem and an expert system.
Rules can represent relations, recommendations, directives, strategies and heuristics:

- **Relation**
  IF the ‘fuel tank’ is empty
  THEN the car is dead

- **Recommendation**
  IF the season is autumn
  AND the sky is cloudy
  AND the forecast is slightly rain
  THEN the advice is ‘take an umbrella’

- **Directive**
  IF the car is dead
  AND the ‘fuel tank’ is empty
  THEN the action is ‘refuel the car’
IF-THEM Rules as a Knowledge Representation

- **Strategy**
  
  IF the car is dead
  THEN the action is ‘check the fuel tank’;
  step1 is complete

  IF step1 is complete
  AND the ‘fuel tank’ is full
  THEN the action is ‘check the battery’;
  step2 is complete

- **Heuristic**
  
  IF the spill is liquid
  AND the ‘spill pH’ < 6
  AND the ‘spill smell’ is vinegar
  THEN the ‘spill material’ is ‘acetic acid’

- Expert systems can also use mathematical operators to define an object as numerical and assign it to the **numerical value**.
  
  IF ‘age of the customer’ < 18
  AND ‘cash withdrawal’ > 1000
  THEN ‘signature of the parent’ is required
Another Way to Represent the Knowledge Base

• Case-based
  • It retrieves cases relevant to the present problem situation from the case base and decides on the solution to the current problem on the basis of the outcomes from previous cases.
  • A case-based ES consists of
    – A case base
    – A retriever
    – An adapter
    – A refiner
    – An executer
    – An evaluator
• A case base functions as a repository of prior cases
  – The cases are indexed so that they can be quickly recalled when necessary
  – A case contains the general descriptions of old problems
Another Way to Represent the Knowledge Base

- **A retriever**: when a new problem is entered into a case based system:
  - A retriever decides on the features similar to the stored cases.
  - Retrieval is done by using features of the new cases as indexes into the case base.
- An **adapter** examines the differences between these cases and the current problem using a similarity function.
  - It then applies rules to modify the old solution to fit the new problem.
- A **refiner** critiques the adapted solution against prior outcomes
  - One way to do this is to compare it to similar solutions of prior cases.
  - If a known failure exists for a derived solution, the system then decides whether the similarities is sufficient to suspect that the new solution will fail.
- **Executor**: Once a solution is critiqued, an executer applies the refined solution to the current problem.
- **Evaluator**: If the results are as expected, no further analysis is made, and the cases and its solution is stored for use in future problem solving.
- If not, the solution is repaired.
The CBR Cycle

- New Case
- Retrieved Case
- Solved Case
- Tested/Repaired Case
- General Knowledge
- Database

Steps:
- Retrieve
- Reuse
- Revise
- Retain/Save
Which Representation do we use?

- **When to use Rule-Based Reasoning?**
  - When there is a lot of specific expert knowledge on a particular subject and the expert can solve the problem sequentially.
  - When an explanation or an audit trail of the solution is required.

- **When to use Case-Based Reasoning?**
  - When the user wants to browse similar cases.
  - When you have lots of typical situations or cases for the knowledge base.
• One of the most powerful parts of an expert system is being able to create rules which state that an answer is probably, but not definitely, true.

• Process accomplished by assigning confidence modes.

• Ways to assign confidence modes:
  – 0-10
  – -100 to 100
  – Increment/decrement system
  – Custom formula
Tree Structured Rules

- **Model:** items of a desk to include a telephone, a computer, a pencil, legal paper, and floppy disks.
Converting Tree Structure to an Expert System

- Rule 1: IF the item makes noise AND the color is black
  - THEN telephone - Confidence 9/10.
- Rule 2: IF the item makes noise AND the color is white
  - THEN computer - confidence 10/10.
- Rule 3: IF the item does not make noise AND the color is black
  - THEN floppy disk - confidence 10/10.
- Rule 4: IF the item does not make noise AND the color is yellow AND the shape is rectangular.
  - THEN legal paper - confidence 7/10.
- Rule 5: IF the item does not make noise AND the color is white AND the shape is cylindrical.
  - THEN pencil - confidence 8/10.
Inference Engine Control Strategies

- In a rule-based expert system, the domain knowledge is represented by a set of IF-THEN production rules and data is represented by a set of facts about the current situation.

- Control Strategies for Execution of Rules
  - Forward Chaining
    - Starts with the facts, and sees what rules apply (and hence what should be done) given the facts.
  
  - Backward Chaining
    - Begins with a goal and works backwards towards the initial conditions will help in answering it.
The inference engine compares each rule stored in the knowledge base with facts contained in the database. When the IF (condition) part of the rule matches a fact, the rule is **fired** and its THEN (action) part is executed.
An example of an Inference Chain

Rule 1: IF $Y$ is true
      AND $D$ is true
      THEN $Z$ is true

Rule 2: IF $X$ is true
      AND $B$ is true
      AND $E$ is true
      THEN $Y$ is true

Rule 3: IF $A$ is true
      THEN $X$ is true
1. The reasoning starts from the known data and proceeds forward with that data.
2. Each time only the topmost rule is executed.
3. When fired, the rule adds a new fact in the database.
4. Any rule can be executed only once.
5. The match-fire cycle stops when no further rules can be fired.

---

**Forward Chaining (Data-Driven) Reasoning**

**Knowledge Base**
- \( Y \& D \rightarrow Z \)
- \( X \& B \& E \rightarrow Y \)
- \( A \rightarrow X \)
- \( C \rightarrow L \)
- \( L \& M \rightarrow N \)

**Database**
- Initial: \( A, B, C, D, E, X \)
- After cycle 1: \( A, B, C, D, E, X, L \)
- After cycle 2: \( A, B, C, D, E, X, L, Y \)
- After cycle 3: \( A, B, C, D, E, X, L, Y, Z \)

**Match**
- Cycle 1: \( X \)
- Cycle 2: \( L \)
- Cycle 3: \( X \)

**Fire**
- Cycle 1: \( Y \& D \rightarrow Z \)
- Cycle 2: \( X \& B \& E \rightarrow Y \)
- Cycle 3: \( Y \& D \rightarrow Z \)
Forward Chaining

- Forward chaining is a technique for gathering information and then inferring from it whatever can be inferred.
- However, in forward chaining, many rules may be executed that have nothing to do with the established goal.
- Therefore, if our goal is to infer only one particular fact, the forward chaining inference technique would not be efficient.
1. The expert system has the goal and the inference engine attempts to find the evidence to prove it.
2. First, the knowledge base is searched to find rules that might have the desired solution.
3. Such rules must have the goal in their THEN (action) parts.
4. If such a rule is found and its IF (condition) part matches data in the database, then the rule is fired and the goal is proved.
5. Thus the inference engine puts aside the rule it is working with (the rule is said to stack) and sets up a new goal, a sub goal, to prove the IF part of this rule.
6. Then the knowledge base is searched again for rules that can prove the sub goal.
7. The inference engine repeats the process of stacking the rules until no rules are found in the knowledge base to prove the current sub goal.
Backward Chaining

**Pass 1**

**Database**

```
A  B  C  D  E  
```

**Knowledge Base**

```
Y & D → Z  
X & B & E → Y  
A → X  
C → L  
L & M → N  
```

**Goal:** Z

**Match**

```
X  
```

**Fire**

```
```

**Pass 2**

**Database**

```
A  B  C  D  E  
```

**Knowledge Base**

```
Y & D → Z  
X & B & E → Y  
A → X  
C → L  
L & M → N  
```

**Sub-Goal:** Y

**Match**

```
Y  
```

**Fire**

```
```

**Pass 3**

**Database**

```
A  B  C  D  E  
```

**Knowledge Base**

```
Y & D → Z  
X & B & E → Y  
A → X  
C → L  
L & M → N  
```

**Sub-Goal:** X

**Match**

```
X  
```

**Fire**

```
```

**Pass 4**

**Database**

```
A  B  C  D  E  
```

**Knowledge Base**

```
Y & D → Z  
X & B & E → Y  
A → X  
C → L  
L & M → N  
```

**Sub-Goal:** X

**Match**

```
Y  
```

**Fire**

```
```

**Pass 5**

**Database**

```
A  B  C  D  E  
```

**Knowledge Base**

```
Y & D → Z  
X & B & E → Y  
A → X  
C → L  
L & M → N  
```

**Sub-Goal:** Y

**Match**

```
Z  
```

**Fire**

```
```

**Pass 6**

**Database**

```
A  B  C  D  E  
```

**Knowledge Base**

```
Y & D → Z  
X & B & E → Y  
A → X  
C → L  
L & M → N  
```

**Goal:** Z

**Match**

```
X  
```

**Fire**

```

```

---

DR. Tarek Helmy@KFUPM-ICS
How do we choose between forward and backward chaining?

- If an expert first needs to gather some information and then tries to infer from it whatever can be inferred, choose the forward chaining inference engine.

- However, if the expert begins with a hypothetical solution and then attempts to find facts to prove it, choose the backward chaining inference engine.
If we have a KB with these 3 rules:

- **Rule 1:**
  IF the ‘traffic light’ is green
  THEN the action is go

- **Rule 2:**
  IF the ‘traffic light’ is red
  THEN the action is stop

- **Rule 3:**
  IF the ‘traffic light’ is red
  THEN the action is go

- Rule 2 and Rule 3, with the same IF part.
- Thus both of them can be set to fire when the condition part is satisfied.
- These rules represent a conflict set.
- The inference engine must determine which rule to fire from such a set.
- A method for choosing a rule to fire when more than one rule can be fired in a given cycle is called **conflict resolution**.
Methods used for Conflict Resolution

- Fire the rule with the **highest priority**. In simple applications, the priority can be established by placing the rules in an appropriate order in the knowledge base.

- Fire the **most specific rule**. This method is also known as the **longest matching strategy**. It is based on the assumption that a specific rule processes more information than a general one.

- Fire the rule that uses the **data most recently entered**. This method relies on time tags attached to each fact in the database. In the conflict set, the expert system first fires the rule whose antecedent uses the data most recently added to the database.
Meta-Knowledge

- Meta-knowledge can be simply defined as knowledge about knowledge. Meta-knowledge is knowledge about the use and control of domain knowledge in an expert system.

- In rule-based expert systems, meta-knowledge is represented by meta-rules. A meta-rule determines a strategy for the use of task-specific rules in the expert system.

- **Meta-rule 1:** Rules supplied by experts have higher priorities than rules supplied by novices.

- **Meta-rule 2:** Rules governing the rescue of human lives have higher priorities than rules concerned with clearing overloads on power system equipment.
Building an Expert System

- Analysis
- Specification
- Development
- Deployment
Analysis and Specification

• **Analysis**
  • **User/expert**: Identify a potential application.
  • **Knowledge Engineer**: Is expert system the answer?
    – Task is well understood
    – Expertise exists, is reliable, and the solution is generally agreed upon.
    – Task is not too hard (but not too easy, either)

• **Specification**
  • User/Expert and Knowledge Engineer work together to define the objectives of the expert system application:
    – Inputs
    – Outputs
    – Methodology
Development

- **Knowledge Engineer**: Learns how the expert performs task:
  - Called “knowledge acquisition”.
  - Must cover current, historical, and hypothetical cases.
- Develop a conceptual model of the ES.
  - Framework consists of high-level descriptions of the tasks and situations.
- Decide how the inference, representation, and control structure can be used to replicate the decision process.
- Build the knowledge base.
- Verify and validate.
  - Am I building the product right?
  - Am I building the right product?
Deployment

- Install the system for routine use.
- Fix bugs, update, and enhance.

Go back to development phase.

This loop remains active throughout the life cycle of the project.
### Expert Systems vs Conventional Programs

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Conventional Program</th>
<th>Expert System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control by...</td>
<td>Information &amp; control integrated</td>
<td>Knowledge separate from control</td>
</tr>
<tr>
<td>Solution by...</td>
<td>Algorithm</td>
<td>Rules &amp; inference</td>
</tr>
<tr>
<td>Execution</td>
<td>Generally sequential</td>
<td>Opportunistic rules</td>
</tr>
<tr>
<td>Input</td>
<td>Must be complete</td>
<td>Can be Incomplete</td>
</tr>
<tr>
<td>Design</td>
<td>Structured</td>
<td>Little or none</td>
</tr>
<tr>
<td>Expansion</td>
<td>Done in major jumps</td>
<td>Incremental</td>
</tr>
<tr>
<td>Representation</td>
<td>Numeric</td>
<td>Symbolic</td>
</tr>
<tr>
<td>Modify</td>
<td>Difficult to modify</td>
<td>Easy to modify</td>
</tr>
<tr>
<td>Results</td>
<td>Final result given</td>
<td>Recommendation</td>
</tr>
<tr>
<td>Interface</td>
<td>Command interface</td>
<td>Natural dialogue</td>
</tr>
<tr>
<td>Answers</td>
<td>Optimal Solution</td>
<td>Acceptable solution</td>
</tr>
</tbody>
</table>
Application Categories

- Interpretation: Inferring situation from observations/data
- Prediction: Inferring likely consequences of situation
- Diagnosis: Inferring malfunctions
- Design: Configuring objects under constraints
- Planning: Developing plans to achieve goals
- Monitoring: Comparing observations to plans
- Debugging: Prescribing remedies for malfunctions
- Repair: Executing a plan to administer a remedy
- Instruction: Diagnosing, debugging, and correcting student performance
- Control: Managing system behavior
- Selection: Identifying best choice from a list of possibilities
- Simulation: Modeling the interaction between system components
Prominent Expert Systems

- **EXSYS** – Provides an easy to use user interface to develop traditional applications or web-based solutions.
- **MYCIN** – Used to diagnose infectious blood diseases and recommend antibiotics.
- **DENDRAL** – Embedded a chemist’s knowledge of mass spectrometry rules to use in analysis.
- **CADUCEUS** – Used to analyze blood-borne infectious bacteria
- .......

DR. Tarek Helmy@KFUPM-ICS
A Case Study Expert System

MYSIN

Medical Diagnoses
Background of MYCIN

- Developed at Stanford University to aid physicians in diagnosing and treating patients with infectious blood diseases.
- Diseases caused by:
  - Bacteria - bacteria in the blood
  - Meningitis - bacterial disease that causes inflammation of the membrane surrounding the brain and spinal cord.
- Rule based system that uses backward chaining
- A team of 20 members worked for several years
Primary Motivating Factors for Building MYCIN

- An expert was required to solve the problem.
- Experts on the problem were scarce or unavailable because of time constraints.
- Immediate expertise was needed in a possibly life threatening situation.
- Time constraints required decisions to be made with limited or inexact information.
- The computer solution needed to be accommodating to the user who may have limited experience with computers.
- Existing solutions may be unreasonable in cases where drug recommendations were inappropriate for the problem.
- Remembering the appropriateness and possible contraindications of a large number of drugs was a challenge for the physician.
Major Features of MYCIN

• Utilizes a backward-chaining system
  ➢ To prove a conclusion of a rule, MYCIN works backward through other rules that support each premise.
  ➢ Other rules provide evidence by working with information obtained from clinical observations or test results.

• Separates knowledge from control
  – Allowed the development team to easily modify knowledge
  – Adding new knowledge only required updating the knowledge base.

• Incorporates meta-rules: Used to redirect the search and allows the system to search intelligently.

EX:  IF The infection is a pelvic-abscess
     AND There are rules that mention in their premise Enterobacteriaceae
     AND There are rules that mention in their premise gram positive rods
     THEN There is suggestive evidence that the rules dealing with Enterobacteriaceae should be evoked before those dealing with gram positive rods
Major Features of MYCIN

- Employs inexact reasoning
  - Physician can respond to a question with UNKNOWN
  - If information is known but uncertain, physician can use a certainty factor reflecting degree of belief (.7 true)
  - Can also use certainty factors in rules to reflect uncertainty IF Evidence THEN Conclusion CF .7.
- Remembers prior session
  - Includes data provided by patient, conclusions drawn by MYCIN, and findings
  - MYCIN can add new information at a later date to update previous UNKNOWN answers.
- Accommodates the user
  - Is easy to use and presents itself in a natural manner
  - Uses natural language interaction - interacts in English
- Has spell checker
  - Validates input for spelling and errors
  - Asks physician to correct spelling
  - Provides user with a list of valid possible legal answers
Major Features of MYCIN

- Provides Explanations
  - Explains *why* and *how* does it derive a conclusion?
  - Explains *why* does it find other results doubtful?

- Provides alternative recommendations
  - Allows physician to choose those drugs from the list of answers he would like alternatives for.
  - Formulates a new drug list, compares it to the other one and presents comparison.
What is Expert System/Decision Support System?

- **Expert System:** Decision-making and/or problem-solving package that can reach a level of performance comparable to (or even exceeding that of) a human expert in some specialized and usually narrow problem area.

- **Decision Support System:** Computer-based support systems which help decision makers utilize data and models to solve semi-structured or unstructured problems.
Summary

• Expert system strengths:
  – Corporate knowledge retained
  – Knowledge can be incomplete -- the expertise can be expanded as needed. Conventional programs must be “complete” before they can be used.
  – Expert systems can act as consultant, instructor, or partner/colleague.

• Expert System limitations:
  – Domain must often be restricted or narrow.
  – Determination of expertise reliability and completeness is very difficult.
  – Some knowledge doesn’t translate well to rules/cases.
  – Expert Systems are expensive -- problem must be sufficiently complex to justify the cost.
The End!!

Thank you

Any Questions?