

# Expert System

- Rule syntax:  
If \_\_\_  
and \_\_\_  
and \_\_\_  
...  
then \_\_\_
- Rule base: ordered list of rules
- Rule-based system
  - Inference engine: tells you how to make decisions
  - Rule base: tells you what to make the decisions with
- Advantages and disadvantages
  - Plus: you can change the rules without changing the engine
  - Minus: this is not a natural format for all types of problems

# Expert System Architecture

- Components
  - Rule base (list of rules)
  - Working memory
- Basic processing loop
  - Start at top
  - Test every rule—see which apply
  - Conflict resolution: if more than one apply, choose one
  - Apply the rule

# More on Architecture

- How to test a rule
  - See if *if*-component is true
- How to apply a rule
  - Do actions in *then*-component
  - Change a value in working memory
  - Output something
  - Ask the user a question  
(input something into working memory)
  - Cannot tell system what to do next!
- Conflict resolution options
  - Take first match
  - Take last match
  - Take closest match (must define)
  - etc.

# MYCIN

- Shortliffe, 1970's (USA)
- Which antibiotic for specified bacterial infection?
- Input: age and sex of patient, symptoms, lab tests
- System asks further questions and requests further lab tests as needed
- Output: what kind of bacteria is suspected and which antibiotic to use
- Sample rule:  
If infection type is \_\_\_  
and suspected entry point is \_\_\_  
and site of culture is \_\_\_  
then bacterium is likely to be \_\_\_

# Dipmeter Advisor

- Schlumberger Corp., late 1970's
- Where should company dig their next oil well?
- Input: from geological testing
- Output: which locations are more likely to have oil
- Sample rule:  
If there is a red pattern over a fault  
and the direction is perpendicular to  
the fault  
and the length is greater than 200 feet  
then the fault is a growth fault

# XCON (= R1)

- DEC Corporation and Carnegie-Mellon University, late 1970's
- Where should each component of a computer be placed in the box?
- Input: details about components
- Output: Organization of computer
- XCON has stages:
  - Check for missing or mismatched components
  - ...
  - Floor plan
  - Cabling
- Stages can be implemented with a flag (state variable) in working memory

# XCON Sample Rules

- If we are in the floor plan stage and there is space for a power supply and there is no power supply available then add a power supply to the order.
- If we are in the floor plan stage and there is space for a power supply and there is a power supply available then put the power supply in the correct place.
- We only choose one rule each time through the loop, so if the first rule is used, we won't use the second rule until the next time through.

# Bagger

- Which items go in which grocery bag?
- Bagger is a toy system, i.e., it is important for pedagogical purposes rather than because we need a computer to help us bag groceries
- Input: items purchased and their qualities
- Output: which items go in which bag
- See Winston article for details
- See separate file for trace of in-class demo
- Features to notice about Bagger
  - Which system(s) have we seen each of these features in?
  - World knowledge encoded in rules
  - Use of stages
  - Ability to add items to input

# Evaluation of Expert Systems

- They worked on a technical level but didn't pay off in a practical sense
  - Cost of building system might have been OK as a one-time cost
  - But cost of keeping the rule base up-to-date make them impractical
- They were brittle
  - = not robust
  - = failed hard
  - When something unexpected happened, they couldn't say anything at all
  - Or duplicate rule, missing rule, etc.
  - Human beings are more flexible
- Were expert systems intelligent?
  - They could solve the problem...