CPSC 322
Introduction to Artificial Intelligence

November 15, 2004

Things...

Term project is due two weeks from today

The final exam will be at noon on Friday, December 10, in MCML 166

Tests and actions

Not limited to troubleshooting -- you could write a "program" of tests and actions to play tic-tac-toe:

```
     (this is the improved version)
if   you occupy two squares in the same row/column/diagonal and
     the third square in that row/column/diagonal is empty
then put your token in that third square

if   opp't occupies two squares in the same row/column/diagonal and
     the third square in that row/column/diagonal is empty
then put your token in that third square

if   the center square is empty
then put your token in the center square

if   a corner square is empty
then put your token in that corner square

if   any square is empty
then put your token in that empty square
```

Tests and actions

Test-action pairs go by many other names:

- if-then rules
- left-hand-sides and right-hand-sides
- antecedent-consequent pairs

Systems that employ test-action pairs are called:

- rule-based systems
- production systems (rules are called productions)
- expert systems (when they're really smart)

Rule-based systems

Typically have three parts:

the rule base

- knowledge encoded as if-then rules
- rules are modular and independent
- each rule has as many tests and actions as necessary
- rules are written in a high-level “rule language” to be interpreted by a “rule interpreter”

Rule-based systems

Typically have three parts:

the working memory or data base

- knowledge of the current state of the world
  - facts
  - goals
  - partial solutions
- this knowledge is changed by application of rules from the rule base

Rule-based systems

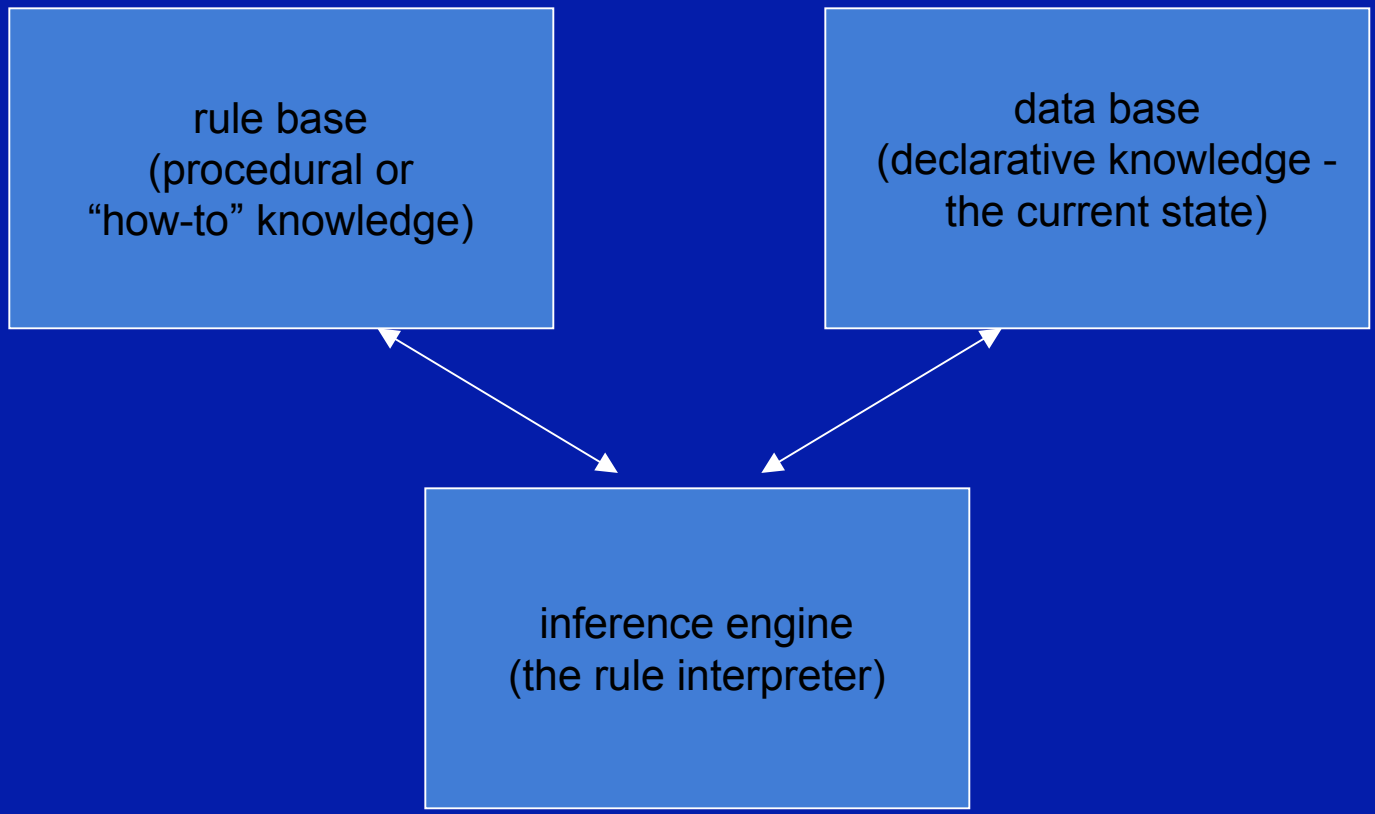
Typically have three parts:

the rule interpreter or inference engine

- defines a language for writing rules
- applies rules to working memory so as to change or update working memory in the following way....

Rule-based systems

Typically have three parts:



Rule-based systems

The inference engine algorithm is simple:

```
until (no tests are true) or
      (some goal has been reached)

  repeat
    1. go through the rule base and collect all rules
       whose tests (left-hand sides) are true (these
       rules are said to be “triggered”)
    2. select one rule to execute
    3. perform the actions (right-hand side) of the
       selected rule (this rule is said to have
       “fired”)
  end repeat
```

Rule-based systems

What if more than one rule is triggered?

```
until (no tests are true) or
      (some goal has been reached)

  repeat
    1. go through the rule base and collect all rules
       whose tests (left-hand sides) are true (these
       rules are said to be “triggered”)
    2. select one rule to execute
    3. perform the actions (right-hand side) of the
       selected rule (this rule is said to have
       “fired”)
  end repeat
```

Rule-based systems

Apply conflict resolution strategy. For example:

- rule ordering - fire the first rule found
- priority - assign priorities to rules in advance
- specificity - fire the rule that has the most tests
- recency - fire the rule that was fired most recently
- frequency - fire the most often used rule
- random - just pick one
- parallel - why settle for one rule? fire them all!

Rule-based systems

Representing knowledge as rules has advantages...

modularity:

Changing one rule may affect overall performance but will not affect the operation of other rules directly. Rules never call other rules; they're independent. So changes to one rule don't require rewriting of other rules.

Rule-based systems

Representing knowledge as rules has advantages...

incremental:

Since rules are independent pieces of knowledge, a rule base can grow incrementally. This enables a system to change and enhance its own expertise by adding, modifying, or deleting rules...such systems exhibit a form of learning.

Rule-based systems

Representing knowledge as rules has advantages...

uniformity:

The rule interpreter or inference engine enforces a uniform representation of knowledge in a particular rule language.

Rule-based systems

Representing knowledge as rules has advantages...

naturalness:

“What to do when...” kinds of knowledge are easily encoded as rules, and the rules are (usually) easily understood by people.

Rule-based systems

Representing knowledge as rules has advantages...

psychologically plausible*:

Some people think that rule-based systems are good models of human problem-solving ability. Analogies are drawn between the rule base and human long-term memory, as well as between the data base (working memory) and human short-term memory.

*The words “psychologically plausible” are often used by people to make their systems sound more “valid” than they are. Sometimes “psychologically plausible” means no more than “We’re not aware of anything that says this model isn’t psychologically plausible, but we haven’t looked very hard.”

Rule-based systems

Representing knowledge as rules has advantages...

spontaneity:

Production systems allow unplanned but useful interactions which are not possible with control structures in which all procedure interactions are determined beforehand. A piece of knowledge can be applied whenever appropriate, not just whenever a programmer predicts it can be appropriate.

from Patrick Winston's "Artificial Intelligence"

Rule-based systems

...but there are some difficulties too...

inefficient -- especially in evaluating tests in left-hand sides of rules; finding the right rules in a rule base is not unlike CILOG or Prolog finding the right rules

opaque -- it's hard to see what the flow of control is (again like logic programming)

adequacy -- can all knowledge be represented as if-then rules?

availability -- how can we get the knowledge that we intend to encode as if-then rules?

Expert systems

As noted in the movie, some people saw rule-based systems as a means of creating powerful but narrowly-scoped AI tools.

The rules in these systems contain lots of highly-specialized domain-specific knowledge for some real but very narrow domain. Expert systems exhibit performance near that of a human expert.

AI's big commercial success, but now treated by many AI people as an unwanted guest.

Expert systems

Just some of many successful applications so far:

- finding organic molecular structure from mass spectrogram (Dendral)
- locating oil and mineral deposits (Prospector)
- medical diagnosis (Mycin, Internist,...)
- computer system component selection (Xcon)
- automobile diagnosis and repair (SBDS)
- training aircraft maintenance personnel (F-16 Maintenance Skills Tutor)

Expert systems

In the big AI boom of the 1980s, “knowledge engineering” firms would build expert system applications for you, for a price.

- provide the expert system shell (a rule interpreter and a language for writing the rules)
- provide expertise in extracting knowledge from selected human experts
- encode the human knowledge as rules

Now, most firms just sell you the shell and you do the rest (that is, you do the hard part)

Finding organic molecular structure

Given a mass spectrogram of an organic compound, you can infer a chemical composition. The hard part is figuring out the molecular structure.

Dendral used several rule-based systems to find the molecular structure.

Finding organic molecular structure

1) use mass spectrogram data to create lists of required and forbidden substructures

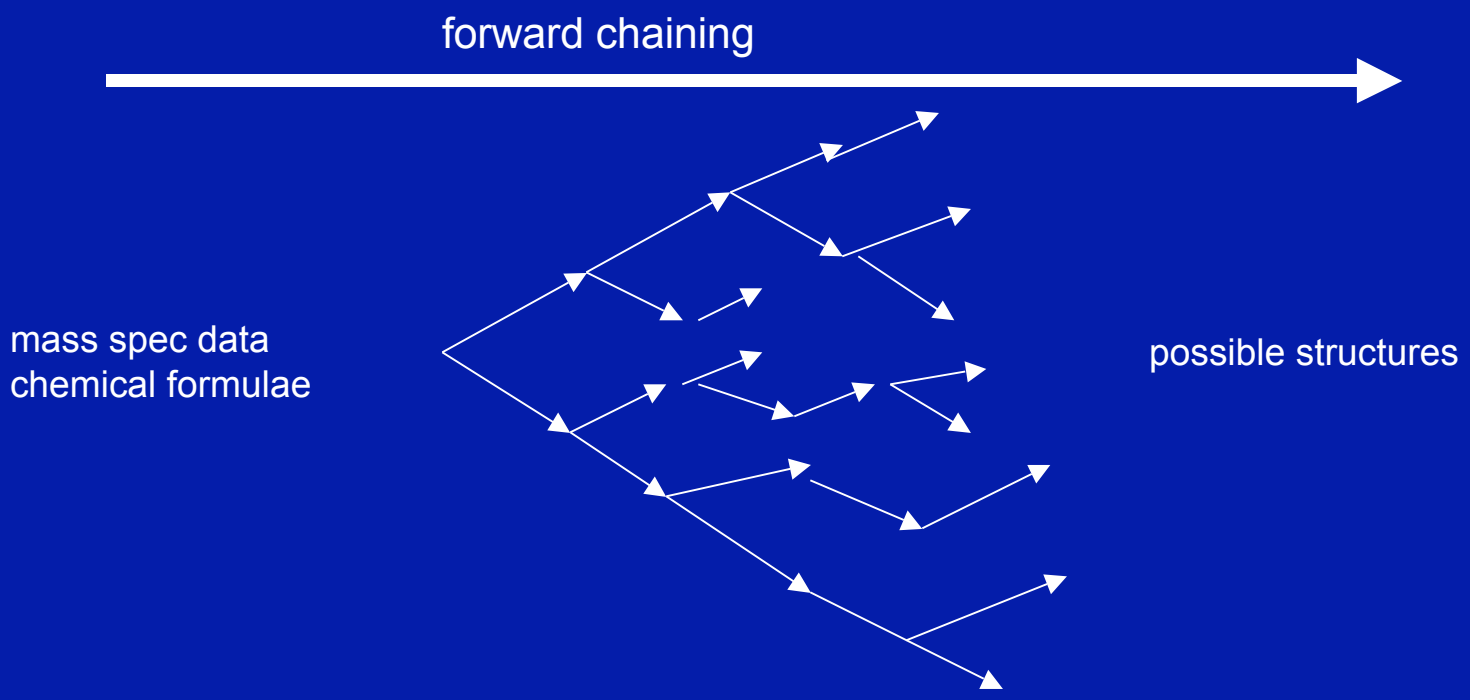
2) use chemical composition formulae to generate all possible structures, then prune using info from step 1

3) generate predicted mass spectrogram data for each remaining proposed structure from step 2

4) find best (possibly partial) match between predicted spectrograms and actual input data

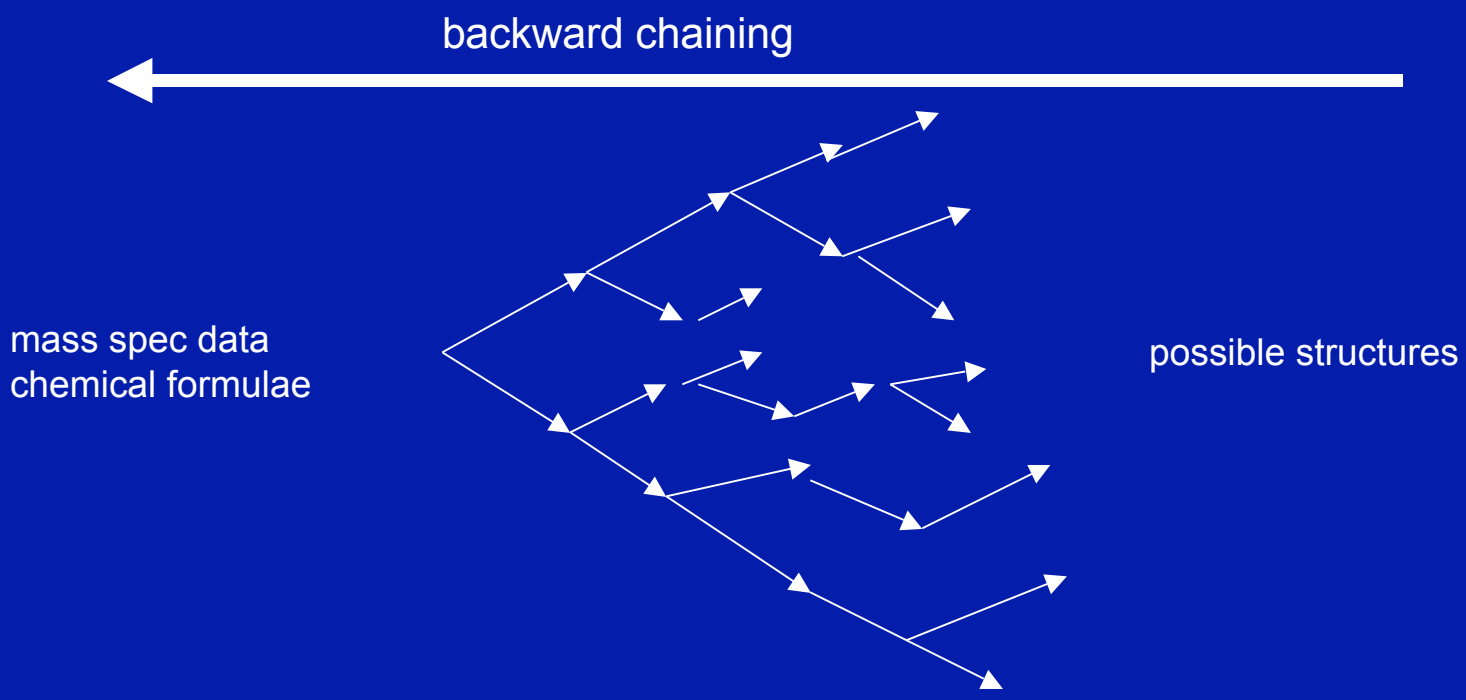
Finding organic molecular structure

Dendral used “forward chaining”, reasoning from input data (start state) to find structures (goal states)



Finding organic molecular structure

Dendral could have used “backward chaining” from all possible structures, but that’s not very efficient



Diagnosing infectious bacterial disease

Mycin was the first successful medical expert system. It had about 300 rules covering about 100 different infectious bacterial diseases.

Mycin built a “start state” by beginning with lots of questions...

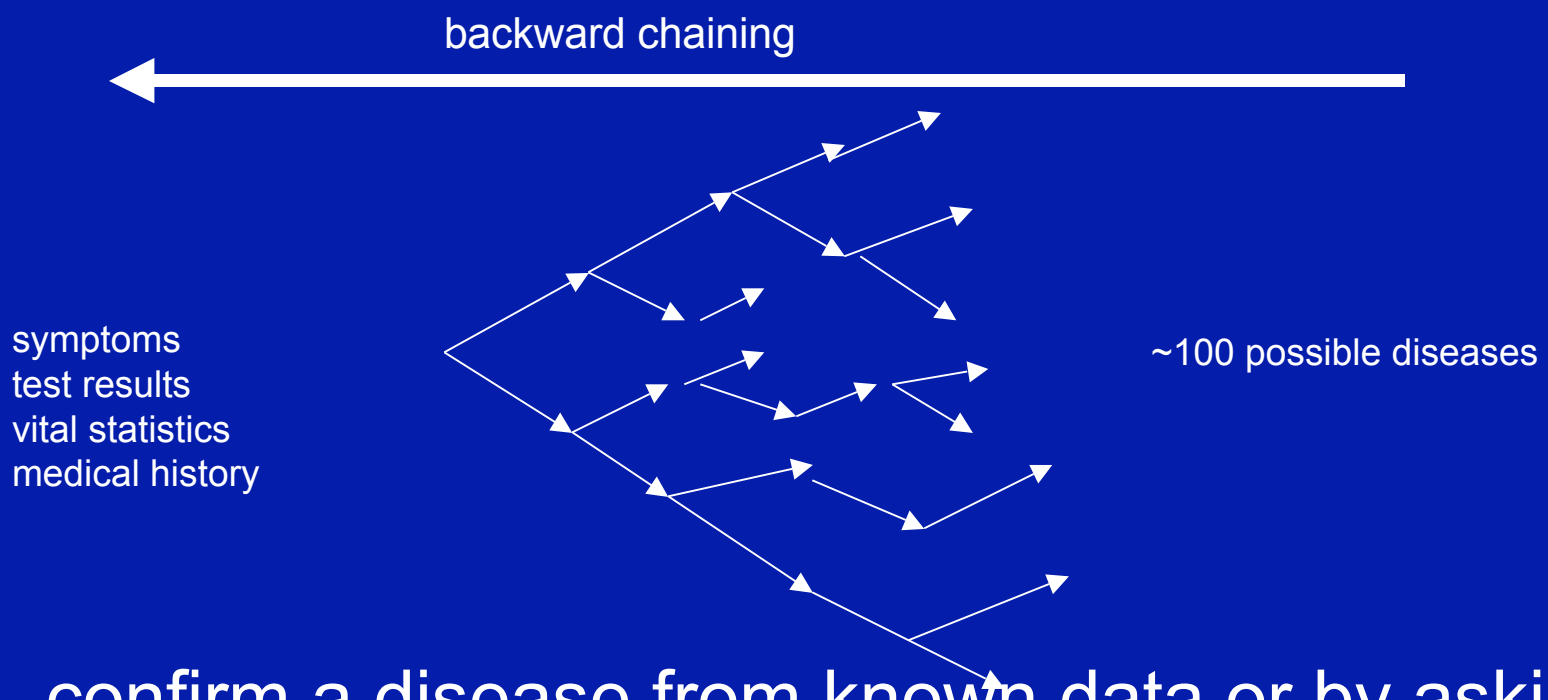
Diagnosing infectious bacterial disease

- What is the patient's name?
  John Doe.
- Male or female?
  Male.
- Age?
  55.
- Have you obtained positive cultures indicating general type?
  Yes.
- What type of infection is it?
  Primary bacteremia.
- When did symptoms first appear?
  May 5.
- Let's call the most recent positive culture C1. From what site was C1 taken?
  From the blood.
- When?
  May 9.
- Let's call the first significant organism from this culture U1. Do you know the identity of U1?
- No.
  Is U1 a rod or a coccus or something else?
- Rod.

  ...and so on...

Diagnosing infectious bacterial disease

Mycin then used “backward chaining”, working backward from the different diseases, trying to...



...confirm a disease from known data or by asking more questions to obtain more data.

Diagnosing infectious bacterial disease

Mycin could have used forward chaining, but...

- the same symptoms can be caused by lots of different bacteria, so forward chaining would "hop around" in its question-asking...it wouldn't appear to be focused, and doctors would become confused and untrusting.

- because backward chaining in this case allows more focused questions, the natural language component is easier to implement

Diagnosing infectious bacterial disease

Mycin could explain its rules and conclusions in English -- important for trust in expert domains

```
sample rule:

if the stain of the organism is gram-positive
   and the morphology of the organism is coccus
   and the growth conformation of the organism is clumps
then (0.7) the identity of the organism is staphylococcus
```

Diagnosing infectious bacterial disease

Mycin could explain its rules and conclusions in English -- important for trust in expert domains

```
sample conclusion:

My recommendation will be based on the opinion that the
identity of U1 may be
1.  Pseudomonas-aeruginosa
2.  Klebsiella-pneumoniae
3.  E. coli
4.  Bacteroides-fragilis
5.  Enterobacter
6.  Proteus-nonmirabilis

...to cover for items 1, 2, 3, 5, and 6, give gentomycin
using a dose of 119 mg (1.7 mg/kg) q8h IV (or IM) for 10
days.  Modify dose in case of renal failure.  Also, to
cover for item 4, give clindamycin using a dose of...
```

Expert systems can be deceptively smart

They can solve complex problems

They can explain, to some degree, how they arrived at a conclusion or why they asked a question

But despite all the expert knowledge they contain, they don't really understand their domain all that well...

A real expert has a causal model of their domain -- Mycin can't explain how bacteria disrupt the normal function of a living organism

A real expert can look at a problem in different ways -- an expert system has no analogical reasoning

Other expert system issues

Domain-specific knowledge doesn't transfer to other domains

Expert systems lack good old common sense

They can't reason easily about their own operation (they lack meta-knowledge)

The interface between human expert and program is a bottleneck -- how do you know what to ask the expert if you don't already have lots of expertise already?

Social issues

Ease of use

Trust

Who gets blamed if system gives wrong answer?

Why would experts want to reveal their expertise?