MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.034 Artificial Intelligence, Fall 2001
Recitation 2, September 14

Rules Rule Prof. Robert C. Berwick

Agenda

1. Administrivia
2. What is intelligence?
    a. Intelligence = "Knowledge Based System" (KBS) = Knowledge + Search
    b. **The 4 Principles of Knowledge:**
        1. How is *knowledge structured*? (REPRESENTATION)
        2. How do we *recognize alternative knowledge*? (MATCHING)
        3. How do we *SEARCH* among alternatives? (forward and backward chaining)
        4. How do we *control* search (priorities if multiple matches – conflict resolution)
3. Breaking the Chains: Rules rule
    - Forward chaining: add a new fact or rule and see what follows
    - Backward chaining: hypothesize and new fact, then prove it
4. The principles of building a Knowledge Based System (aka "Knowledge Engineering")
5. Applying what you know: an Example

1. Administrivia

Who I am – Bob Berwick; 35-423, x8918, berwick@ai.mit.edu
PS info

2. Simple (trivial) Ideas can be Powerful: Intelligence= Knowledge-Based Systems as Reasoning systems

- Knowledge-based = knowledge + search. *How* to represent knowledge *How* to do search. Important: *YOU CAN'T DO OR UNDERSTAND KNOWLEDGE ENGINEERING BY JUST TALKING ABOUT IT.*
- How to do search: essentially, *logic* (one rule: (Generalized) *modus ponens*) , viz., from an implication and the premise of the implication, you can infer the conclusion. ($\alpha \Rightarrow \beta$ and α, conclude β, e.g., Person implies mortal & Socrates is a person, conclude Socrates is mortal, an *inference*).

    Antecedent (If part): **Person** x
    Consequent (new assertion: **Mortal** x
- But, two different ways of doing this:
- **Forward chaining:** Start with sentences in the knowledge base and generate new conclusions that in turn can allow new inferences to be made. Usually used when we add a new fact to the knowledge database and we want to generate its consequences. ("Top-down" branching). Example: bagger system (add list of stuff to pack, and see what assertions are possible; note that we use *variables* here in matching.)

Forward chaining algorithm:

1. Pick a rule to try
2. Create rule instances where the antecedents of the rules match the assertions
3. If > 1 match, apply conflict resolution strategy to choose rule "then" portion to add assertion
4. Add this assertion, and
5. Repeat until no new assertions can be added (until closure).

- **Backward chaining:** Start with something we want to prove, find implication sentences that would allow us to conclude it, and then attempt to establish their premises in turn. This is normally used when we want a goal to be proved. ("Bottom-up" inference – must use *variables* in *both* the goal and the rules to try *tentative* (guessed) assignments.) Example: zoo world; medical diagnosis.

Pseudo-code for backchaining:

```
Rule form:  IF <premise> THEN <conclusion>
where   <premise> is a list C of clauses.

1. Find the list of rules that conclude about the
current attribute A (unification may be required)

2. If there are no such rules,
ask the user for the value of the attribute A.

3. Otherwise, for each rule R, use the following to determine the truth of
the premise of the rule:

   For each clause C in the premise of R:
      3.1 Look in the database for the value of the
               attribute mentioned in C.

      3.2 If there is no information in the database
          about that attribute,

      [recursion step: take another step back in the inference chain –
create AND/OR Tree]:
      3.3 Start from the top with that attribute as the current one.

      Evaluate the truth of C using the information  in the database

4. If the premise of the rule evalutes to "true"
   make the conclusion shown given in the rule (set the attribute value).
```

Rules r1 through r6:

r1: if FATIGUED & FEVER then *DIAGNOSIS IS COLD*

r2: if MUSCLE-ACHES & NORMAL-TEMP then *DIAGNOSIS IS FLU*

r3: if SPOTTED & ¬IMMUNE then *DIAGNOSIS IS MEASLES*

r4: if SLEEPS & ¬EXERCISE then *FATIGUED*

r5: if TEMP>101 then *FEVER*

r6: if ¬VACCINATED & ¬HAD-MEASLES then ¬IMMUNE

Example of backward chaining: diagnosis example (see rules on slides/handout/recitation). Given: patient walks in. What's the first question Dr. Backchainer asks? (Why not use forward chaining for this?).
Assume that you answer "yes" to all questions except the patient does *not* sleep a lot and the patient's temperature is normal.

```
(backchain-asking-user '(the diagnosis is ?x))

Rule that triggers:
Question asked:
Rule that triggers:
Question asked:
```

- **Comparison of knowledge engineering and programming**

Knowledge Engineering	*Programming*
Choosing a logic	Choosing a programming language
Building a knowledge base	Writing a program
Encoding the reasoning (proof) theory	Choosing or writing a compiler
Inferring new facts	Running a program

3. How to build Knowledge-based (Expert) Systems

Recall *Winnie the Pooh.* Suppose we want to conclude, along with Christopher Robin, that Pooh, being a bear of very small brain, is silly. We might do this:

Proposal 1: Add the following If-then rule and the following assertion

```
If   (?b is BearOfVerySmallBrain)
Then (?b is Silly)

BearOfVerySmallBrain(Pooh)
```

But this is a lousy assertion. From `BearOfVerySmallBrain(Pooh)` one **cannot** conclude that Pooh has a brain at all; or that Pooh is a bear, or that very small brains are smaller than small brains. This is too *specific* a representation if we want to conclude, e.g., *Silly(Pooh).* How can we tell?

Answer: It's *Wrong* because the statement is at *too specific* a level. Adding *AnotherVeryLongName* takes just as much work as adding the first. For example, to conclude *Silly(Piglet)* from *ShyBabyPigOfSmallBrain(Piglet),* what rule do we write? Note that the first fact about silliness is no help in a similar situation. *You should need fewer and fewer new facts as you go along, and fewer new if-then rules!* If you wind up with lots and lots of too specific rules, you've gone astray.

Principle: Each time you write down a rule, you should ask yourself the following questions

- *Why* is this true? Could I write down some facts that make it true instead?
- *How generally* is it applicable? Can I state it for a more general class of objects?
- Do I need a new property (predicate) to denote the objects? How does this class relate to others?

Principle: Write rules at the most *general* level at which the knowledge is applicable. (*Consistent* and *useful* in a world of stuffed animals.) So, what should we do in this case? Let me get you started. How should we tie knowledge about Pooh into a broader context and at the appropriate level of generality? How should we provide a precise sense of "very small"? Is Pooh's brain very small compared to a molecule or the Sun? Start with: Pooh is a bear; bears are animals.

```
(Pooh is a bear)
r1: if   (?x is a bear)
     then (?x is an animal)

r2: if  (relative-size(brain ?x), (brain (typical-member(species-of ?x)))
    then (silly ?x)
```

What about connecting *very* small brains to silliness?

4. The 5-step process in building a KBS, redux

1. **Decide what to talk about: Camembert example**
2. **Decide on a vocabulary of predicates, constants:** translate the domain-level concepts into names (e.g., *size*, *animal,* …)
3. **Encode general knowledge about the domain.** (And debug it)
4. **Encode the specific instances** - a description of the *specific* problem instance
5. **Pose queries to the inference procedure & debug**

```
DIAGNOSIS.K
RULES

(r1
 IF
 (the patient is FATIGUED)
 (the patient has a FEVER)
 THEN
 (the DIAGNOSIS of the patient is COLD))

(r2
 IF
 (the patient has MUSCLE aches)
 (the patient has NORMAL temperature)
 THEN
 (the DIAGNOSIS of the patient is FLU))

(r3
 IF
 (the patient has a spotted RASH)
 (the patient is not IMMUNE to measles)
 THEN
 (the DIAGNOSIS of the patient is MEASLES))

(r4
 IF
 (the patient SLEEPS a lot)
 (the patient avoids EXERCISE)
 THEN
 (the patient is FATIGUED))

(r5
 IF
 (the temperature of the patient is GREATER than 101)
 THEN
 (the patient has a FEVER))

(r6
 IF
 (the patient has not been VACCINATED for measles)
 (the patient has not had measles PREVIOUSLY)
 THEN
 (the patient is not IMMUNE to measles))
```