CSC 550: Introduction to Artificial Intelligence

Spring 2004

Knowledge-based problem solving
- expert systems
  rule-based reasoning, heuristics
  reasoning with uncertainty
  - Bayesian probabilities, certainty factors, fuzzy reasoning
  alternative approaches
  - case-based reasoning, model-based reasoning
- planning
  state space view, frame problem
  STRIPS

Expert systems

expert systems are AI's greatest commercial success

an expert system uses knowledge specific to a problem domain to provide "expert quality" performance in that application area

- DENDRAL (1967)   determine molecular structure based on mass spectrogtrams
- MYCIN (1976)   diagnosis & therapy recommendation for blood diseases
- PROSPECTOR (1978)   mineral exploration (found a $100M ore deposit)
- XCON (1984)   configure VAX and PDP-11 series computer systems
                          (saved DEC $70M per year)

today, expert systems are used extensively in finance, manufacturing, scheduling, customer service, …
- FocalPoint (TriPath Imaging) screens ~10% of all pap smears in U.S. (2002)
- American Express uses an ES to automatically approve purchases
- Mrs. Field's cookies uses an ES to model the founder's operational ideas
- TaxCut uses an ES to give tax advice
- Phoenix Police Dept uses an ES to help identify suspects using M.O.

Common characteristics of expert systems

➢ system performs at a level generally recognized as equivalent to a human expert in the field
  - presumably, human expertise is rare or expensive
  - the demand for a solution justifies the cost & effort of building the system

➢ system is highly domain specific
  - lots of knowledge in a narrow field (does not require common sense)
  - amenable to symbolic reasoning, but not solvable using traditional methods

➢ system can explain its reasoning
  - in order to be useful, it must be able to justify its advice/conclusions

➢ system manipulates probabilistic or fuzzy information
  - must be able to propagate uncertainties and provide a range of conclusions

➢ system allows for easy modification
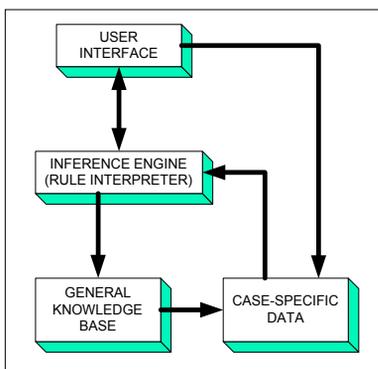 - knowledge bases must be refined

3

System architecture

usually, expert systems are rule-based
  - extract expert knowledge in the form of facts & rules

```
if P1 and P2 and P3, then conclude C.
```


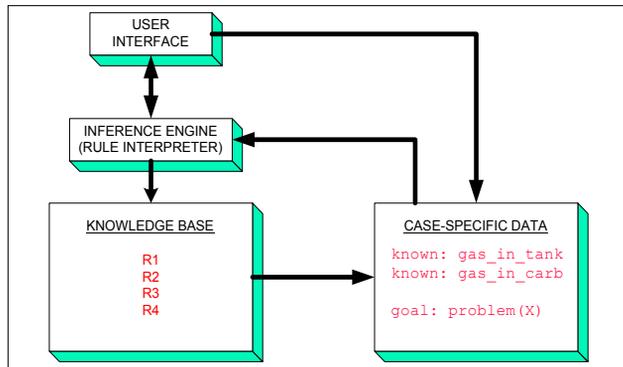|	
user interface	acquires information and displays results
inference engine	performs deductions on the known facts & rules (i.e., applies the knowledge base)
knowledge base	domain specific facts & rules for solving problems in the domain
case-specific data	working memory, stores info about current deduction

4

Inference example

Consider the following rules about diagnosing auto problems

```
(R1) if gas_in_engine and turns_over, then problem(spark_plugs).
(R2) if not(turns_over) and not(lights_on), then problem(battery).
(R3) if not(turns_over) and light_on, then problem(starter).
(R4) if gas_in_tank and gas_in_carb, then gas_in_engine.
```

USER INTERFACE

INFERENCE ENGINE (RULE INTERPRETER)

KNOWLEDGE BASE

R1
R2
R3
R4

CASE-SPECIFIC DATA

known: gas_in_tank
known: gas_in_carb

goal: problem(X)

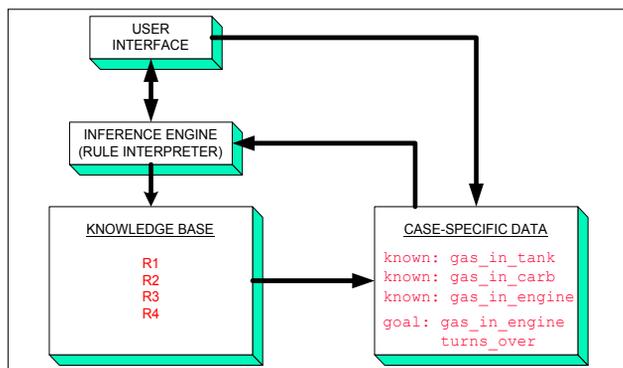Knowledge Base (KB) contains the general rules & facts about the domain

User Interface may be used to load initial facts about the specific task, specify a goal

5

Inference example (cont.)

Consider the following rules about diagnosing auto problems

```
(R1) if gas_in_engine and turns_over, then problem(spark_plugs).
(R2) if not(turns_over) and not(lights_on), then problem(battery).
(R3) if not(turns_over) and light_on, then problem(starter).
(R4) if gas_in_tank and gas_in_carb, then gas_in_engine.
```

USER INTERFACE

INFERENCE ENGINE (RULE INTERPRETER)

KNOWLEDGE BASE

R1
R2
R3
R4

CASE-SPECIFIC DATA

known: gas_in_tank
known: gas_in_carb
known: gas_in_engine
goal: gas_in_engine
      turns_over

Inference Engine can make forward deductions (use rules and existing facts to deduce new facts)

can also reason backwards, reducing goal to subgoals (ala Prolog)

goals can be solved by facts, or may prompt the user for more info

6

Rule-based reasoning

rule-based expert systems have many options when applying rules
- forward reasoning vs. backward reasoning
- depth first vs. breadth first vs. …
- apply "best" rule vs. apply all applicable rules

also, many ways to handle uncertainty
- probabilities
  specify likelihood of a conclusion, apply Bayesian reasoning
- certainty factors
  a certainty factor is an estimate of confidence in conclusions
  not probabilistically precise, but effective
- fuzzy logic
  reason in terms of fuzzy sets (conclusion can be a member to a degree)
  again, not probabilistically precise, but effective

Case study: MYCIN

MYCIN (1976) provided consultative advice on bacterial infections
- rule-based
- backward reasoning (from a specific goal back to known facts)
- performs depth first, exhaustive search of all rules
- utilizes certainty factors

sample rule:
```
IF:   (1) the stain of the organism is gram-positive, AND
      (2) the morphology of the organism is coccus, AND
      (3) the growth confirmation of the organism is clumps,
THEN: there is suggestive evidence (0.7) that
      the identity of the organism is staphylococcus.
```

MYCIN used rules to compute certainty factors for hypotheses
1. find rules whose conclusions match the hypothesis
2. obtain CF's for premises (look up, use rules, ask, …) and compute the CF for the conclusion
3. combine CF's obtained from all applicable rules.

Certainty factors in MYCIN

Consider two rules:

    (R1) hasHair → mammal   CF(R1) = 0.9

    (R2) forwardEyes & sharpTeeth → mammal   CF(R2) = 0.7

Suppose you have determined that:

    CF(hasHair) = 0.8   CF(forwardEyes) = 0.75   CF(sharpTeeth) = 0.3

Given multiple premises, how do you combine into one CF?

    $CF(P1 \lor P2) = max(CF(P1), CF(P2))$

    $CF(P1 \land P2) = min(CF(P1), CF(P2))$

    So, $CF(forwardEyes \land sharpTeeth) = min(0.75, 0.3) = 0.3$

Certainty factors in MYCIN

Consider two rules:

    (R1) hasHair → mammal   CF(R1) = 0.9

    (R2) forwardEyes & sharpTeeth → mammal   CF(R2) = 0.7

We now know that:

    CF(hasHair) = 0.8   CF(forwardEyes) = 0.75   CF(sharpTeeth) = 0.3

    $CF(forwardEyes \land sharpTeeth) = min(0.75, 0.3) = 0.3$

Given the premise CF, how do you combine with the CF for the rule?

    CF(H, Rule) = CF(Premise) * CF(Rule)

    So, CF(mammal, R1)   = CF(hasHair) * CF(R1) = 0.8 * 0.9 = 0.72

        CF(mammal, R2) $= CF(forwardEyes \land sharpTeeth) * CF(R2)$

                               = 0.3 * 0.7

                               = 0.21

Certainty factors in MYCIN

Consider two rules:
    (R1) hasHair → mammal   CF(R1) = 0.9
    (R2) forwardEyes & sharpTeeth → mammal   CF(R2) = 0.7

We now know that:

$CF(hasHair) = 0.8$ $CF(forwardEyes) = 0.75$ $CF(sharpTeeth) = 0.3$
$CF(forwardEyes \wedge sharpTeeth) = min(0.75, 0.3) = 0.3$
$CF(mammal, R1) = 0.72$ $CF(mammal, R2) = 0.21$

Given diff rules with same conclusion, how do you combine CF's?

$CF(H, Rule \& Rule2) = CF(H, Rule1) + CF(H, Rule2)*(1-CF(H,Rule1))$

So, CF(mammal, R1 & R2)
                = CF(mammal, R1) + CF(mammal, R2)*(1-CF(mammal,R1))
                = 0.72 + 0.21*0.28
                = 0.72 + 0.0588
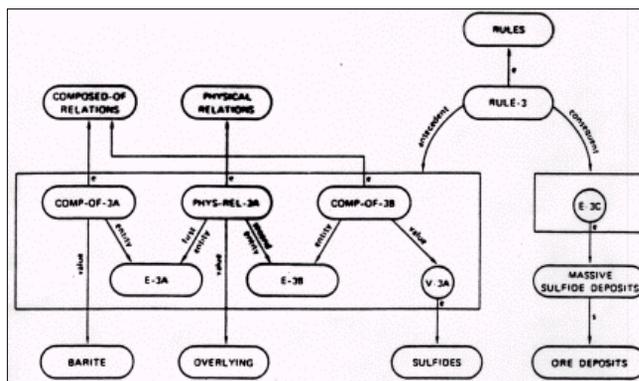   = 0.7788

note: CF(mammal, R1 & R2) = CF(mammal, R2 & R1)

Rule representations

rules don't have to be represented as IF-THEN statements

PROSPECTOR (1978) represented rules as semantic nets
- allowed for inheritance, class/subclass relations
- allowed for the overlap of rules (i.e., structure sharing)
- potential for a smooth interface with natural language systems

Barite overlying sulfides suggests the possible presence of a massive sulfide deposit.

Knowledge engineering

knowledge acquisition is the bottleneck in developing expert systems
- often difficult to codify knowledge as facts & rules
- extracting/formalizing/refining knowledge is long and laborious
  known as *knowledge engineering*

in addition, explanation facilities are imperative for acceptance
- TEIRESIAS (1977)   front-end for MYCIN, supported knowledge
                                    acquisition and explanation

  could answer   WHY is that knowledge relevant
                          HOW did it come to that conclusion
                          WHAT is it currently trying to show

  could add new rules and adjust existing rules

today, expert system shells are a huge market
- ES shell is a general-purpose system, can plug in any knowledge base
- includes tools to assist in knowledge acquisition and refinement

Expert system shell

eXpertise2Go provides a free expert system shell
http://www.expertise2go.com/webesie/e2gdoc/
- the shell provides the inference engine, acquisition/explanation interface
- the user must insert the relevant knowledge base

- e2gLite shell is a Java applet, utilizes very nice GUI interface
  features for knowledge acquisition, explanation, justification, …

Automobile diagnostic system

The result of switching on the headlights is:

○ they light up

○ nothing happens

◉ I don't know/would rather not answer

Very uncertain (50%) ○ ○ ○ ○ ○ ◉   Very certain (100%)

| Submit your response | Why ask? | Restart |

view page in browser

Knowledge base consists of rules…

```
REM Sample knowledge base

RULE [Is the battery dead?]
If [the result of switching on the headlights] = "nothing happens" or
[the result of trying the starter] = "nothing happens"
Then [the recommended action] = "recharge or replace the battery"


RULE [Is the car out of gas?]
If [the gas tank] = "empty"
Then [the recommended action] = "refuel the car"


RULE [Is the battery weak?]
If [the result of trying the starter] : "the car cranks slowly" "the car cranks normally" and
[the headlights dim when trying the starter] = true and
[the amount you are willing to spend on repairs] > 24.99
Then [the recommended action] = "recharge or replace the battery"


RULE [Is the car flooded?]
If [the result of trying the starter] = "the car cranks normally" and
[a gas smell] = "present when trying the starter"
Then [the recommended action] = "wait 10 minutes, then restart flooded car"


RULE [Is the gas tank empty?]
If [the result of trying the starter] = "the car cranks normally" and
[a gas smell] = "not present when trying the starter"
Then [the gas tank] = "empty" @ 90

. . .
```

REM specifies a comment

rules can test values, using:
=   equals
!   not equals
:   equals one of
> < comparisons

rules can specify a certainty factor for the conclusion

15

… and prompts for acquiring info

```
PROMPT [the result of trying the starter] Choice CF
"What happens when you turn the key to try to start the car?"
"the car cranks normally"
"the car cranks slowly"
"nothing happens"

PROMPT [a gas smell] MultChoice CF
"The smell of gasoline is:"
"present when trying the starter"
"not present when trying the starter"

PROMPT [the result of switching on the headlights] MultChoice CF
"The result of switching on the headlights is:"
"they light up"
"nothing happens"

PROMPT [the headlights dim when trying the starter] YesNo CF
"Do the headlights dim when you try the starter with the lights on?"

PROMPT [the gas tank] MultChoice CF
"According to the fuel gauge, the gas tank is:"
"empty"
"not empty"

PROMPT [the amount you are willing to spend on repairs] Numeric CF
"How much are you willing to spend on repairs? (enter value 0->500)"
"0"
"500.0"

GOAL [the recommended action]

MINCF 80
```

different types of prompts:
Choice : pull-down menu
MultiChoice : diff options
YesNo : either yes or no
Numeric : text box for range

CF qualifier adds radio buttons so user can enter certainties

GOAL specifies the top-level goal

MINCF gives the certainty threshold

16

Inference mechanism

the inference engine works backwards from a goal
- stops as soon as finds rule that concludes goal with desired certainty

handles uncertainties via certainty factors (similar to MYCIN)

- associate a CF between 50% (very uncertain) and 100 (very certain) for knowledge

  - each fact and rule in the KB will have a CF associated with it (default 100%)
  - for askable info, the user can specify a CF for that info

  - combine CF's of rule premises as in MYCIN
    $$CF(P1 \vee P2) = max(CF(P1), CF(P2))$$
    $$CF(P1 \wedge P2) = min(CF(P1), CF(P2))$$

  - combine rule premises and conclusion CF as in MYCIN
    $$CF(H, Rule) = CF(Premise) * CF(Rule)$$

  - will only consider a premise or rule if its CF exceeds a threshold (e.g., 80)

For HW4

You are to complete a knowledge base for identifying animals.
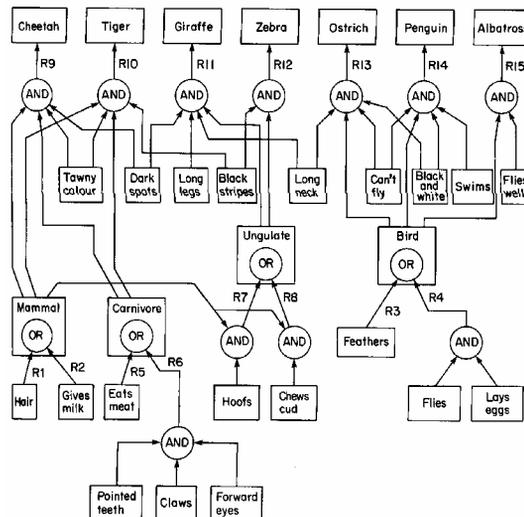
Partial knowledge base

```
REM Animal knowledge base (animal.kb)
REM Author: Dave Reed
REM Date: 3/26/03

RULE [is it a penguin?]
If [animal order] = "bird" and
   [animal color] = "black and white" and
   [type of movement] ! "flies"
Then [animal identification] = "penguin"

RULE [is it an ostrich?]
If [animal order] = "bird" and
   [animal color] = "black and white" and
   [type of movement] ! "flies" and
   [type of neck] = "long"
Then [animal identification] = "ostrich"

RULE [is it an albatross?]
If [animal order] = "bird" and
   [type of movement] = "flies"
Then [animal identification] = "albatross"

RULE [order is bird?]
If [type of surface] = "feathers"
Then [animal order] = "bird"

RULE [order is bird?]
If [type of movement] = "flies" and
   [type of reproduction] = "lays eggs"
Then [animal order] = "bird"

. . .
```

view page in browser

```
PROMPT [type of surface] MultChoice CF
"What kind of skin covering does the animal
have?"
"hair"
"scales"
"feathers"
"smooth skin"

PROMPT [animal color] MultChoice CF
"What color is the animal?"
"tawny"
"black and white"
"none of the above"

PROMPT [type of movement] MultChoice CF
"How does the animal move around?"
"walks"
"swims"
"flies"

PROMPT [type of reproduction] MultChoice CF
"How does the animal reproduce?"
"lays eggs"
"gives birth"
"asexually"

PROMPT [type of neck] MultChoice CF
"How would you describe the animal's neck?"
"long"
"short"
"no neck"

GOAL [animal identification]

MINCF 70
```

Alternative approaches

case-based reasoning
- begin with a collection of cases (previous solutions)
- when you encounter a new situation, find the closest match and modify it to apply to the new situation

common applications: legal advice, hardware diagnosis, help-line, …

model-based reasoning
- attempt to construct a model of the situation
- provides deeper understanding of the system, but more difficult & detailed

common examples: hardware diagnosis
    construct software models of individual components
    when an error occurs, compare with the model's behavior

model-based reasoning is used to troubleshoot NASA space probes

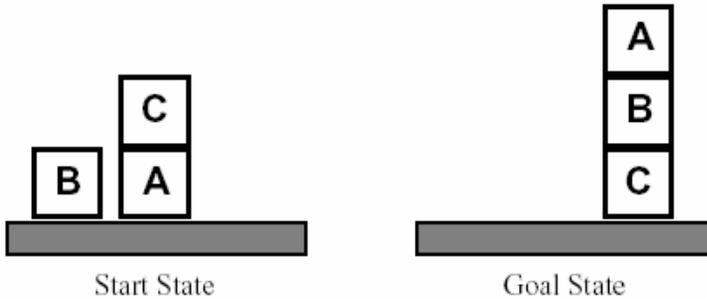see Chapter 7 for summary of advantages/disadvantages

Planning

the subfield of planning combines rule-based reasoning with search

a *planner* seeks to find a sequence of actions to accomplish some task
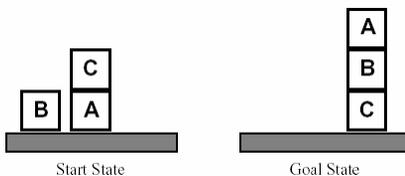- e.g., consider a simple blocks world

Start State   Goal State

Planning as state space search

can describe the environment (state) using predicates:

gripping(Block)	clear(Block)
on(BlockTop, BlockBottom)	onTable(Block)

Start State   Goal State

start state:

gripping()	onTable(A)
onTable(B)	clear(B)
on(C, A)	clear(C)

goal state:

gripping()	onTable(C)
on(B, C)	on(A,B)
clear(A)	

Planning as state space search

similarly, can describe actions using predicates & rules:

pickup(B):   if clear(B) & onTable(B) & gripping() ➜ gripping(B)
putdown(B):   if gripping(B) ➜ onTable(B) & clear(B) & gripping()
stack(T, B):   if gripping(T) & clear(B) ➜ on(T, B) & clear(T) & gripping()
unstack(T, B): if gripping() & on(T, B) & clear(T)➜ clear(B) & gripping(T)

in theory, we could define the entire state space using such rules

- generate a plan (sequence of actions) in the same way we solved flights, jugs, …

in practice, this gets very messy

- the *frame problem* refers to the fact that you not only have to define everything that is changed by an action, but also everything that ISN'T changed

  e.g., when you pickup a block, all other blocks retain same properties

- as complexity increases, keeping track of what changes and doesn't change after each action becomes difficult
- dividing a problem into pieces is tricky, since solution to one might undo other

---

STRIPS approach

STRIPS (STanford Research Institute Planning System)

- approach introduced by Fikes & Nilsson at SRI, 1971
- was used to control a robot that moved around a building, took out trash
- has served as the basis of many planning systems since then

idea: for each action, associate 3 sets

- preconditions for the action, additions & deletions as a result of the action

pickup(B)
    P: {clear(B), onTable(B), gripping()}   **A:** {gripping(B)}   **D:** {onTable(B), gripping()}
putdown(B)
    P: {gripping(B)}   **A:** {onTable(B), clear(B), gripping()}   **D:** {gripping(B)}
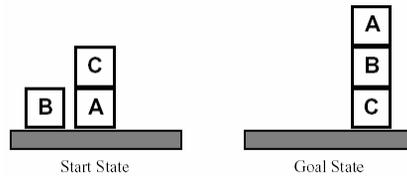stack(T, B)
    P: {gripping(T), clear(B)}   **A:** {on(T, B), clear(T), gripping()}   **D:** {clear(B), gripping(T)}
unstack(T, B)
    P: {gripping(), on(T, B), clear(T)}   **A:** {clear(B), gripping(T)}   **D:** {on(T, B), gripping()}

STRIPS example:

Start State   Goal State

start state: { gripping(), onTable(A), onTable(B), clear(B), on(C, A), clear(C) }

unstack(C, A): { onTable(A), onTable(B), clear(B), clear(A), gripping(C) }

putdown(C):   { onTable(A), onTable(B), clear(B), clear(A), onTable(C), clear(C), gripping() }

pickup(B):   { onTable(A), clear(A), onTable(C), clear(C), gripping(B) }

stack(B, C):   { onTable(A), clear(A), onTable(C), on(B, C), clear(B), gripping() }

pickup(A):   {onTable(C), on(B, C), clear(B), gripping(A) }

stack(A, B):   {onTable(C), on(B, C), on(A, B), clear(A), gripping() }

25

Next week…

Connectionist models of machine learning
- connectionist models, neural nets
- perceptrons, backpropogation networks
- associative memory, Hopfield nets

Read Chapters 9, 10

Be prepared for a quiz on
- this week's lecture (moderately thorough)
- the reading (superficial)

26