CSC 550: Introduction to Artificial Intelligence

Fall 2008

Knowledge representation

- associationist knowledge
  semantic nets, conceptual dependencies
- structured knowledge
  frames, scripts
- expert systems
  rule-based reasoning, heuristics
  reasoning with uncertainty

1

Knowledge representation

underlying thesis of GOFAI: Intelligence requires

- the ability to represent information about the world, and
- the ability to reason with the information

knowledge representation schemes

- logical: use formal logic to represent knowledge
  e.g., state spaces, Prolog databases
- procedural: knowledge as a set of instructions for solving a problem
  e.g., production systems, expert systems (next week)
- associationist: knowledge as objects/concepts and their associations
  e.g., semantic nets, conceptual dependencies
- structured: extend networks to complex data structures with slots/fillers
  e.g., scripts, frames

2

Semantic nets (Quillian, 1967)

main idea: the meaning of a concept comes from the way it is connected to other concepts

SNOW

in understanding language and/or reasoning in complex environments, we make use of the rich associativity of knowledge

When Timmy woke up and saw snow on the ground, he immediately turned on the radio.

3

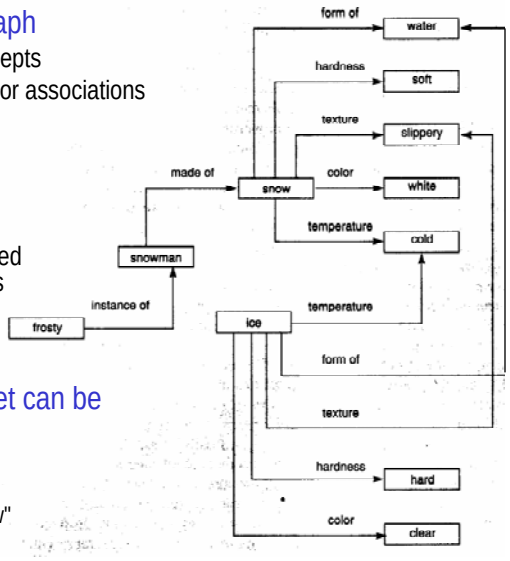
graphs of concepts

can represent knowledge as a graph

- nodes represent objects or concepts
- labeled arcs represent relations or associations

such graphs are known as semantic networks (nets)

- the meaning of a concept is embodied by its associations to other concepts



retrieving info from a semantic net can be seen as a graph search problem

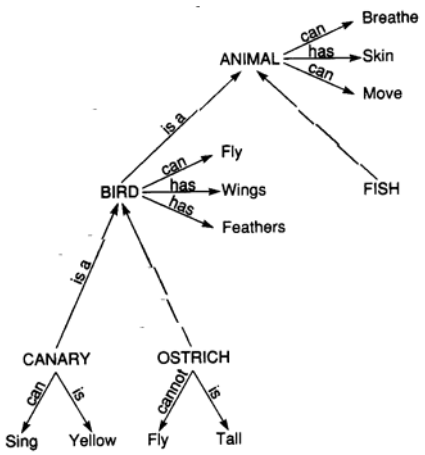
to find the texture of snow

1. find the node corresponding to "snow"
2. find the arc labeled "texture"
3. follow the arc to the concept "slippery"

4

semantic nets & inheritance

in addition to data retrieval, semantic nets can provide for *deduction* using inheritance



since a canary is a bird, it *inherits* the properties of birds (likewise, animals)

e.g., canary can fly, has skin, ...

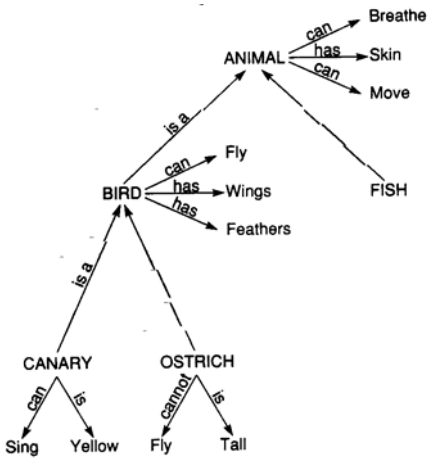
to determine if an object has a property,

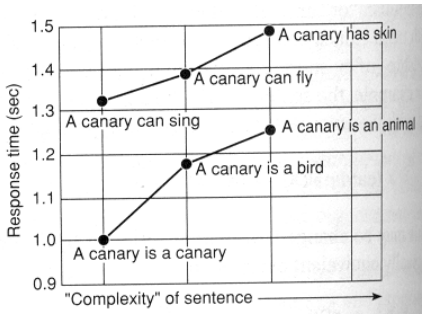
- look for the labeled association,
- if no association for that property, follow *is_a* link to parent class and (recursively) look there

5

Inheritance & cognition

Quillian and Collins (1969) showed that semantic nets with inheritance modeled human information storage and retrieval



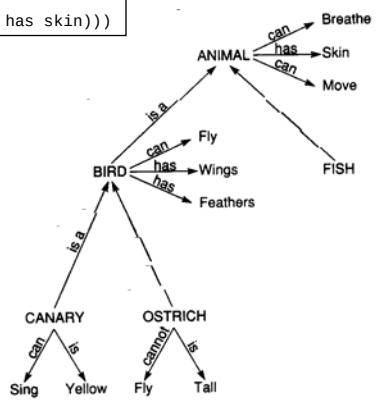


6

Semantic nets in Scheme

can define a semantic net in Scheme as an association list

```
(define ANIMAL-NET
  '((canary can sing) (canary is yellow) (canary is-a bird)
    (ostrich is tall) (ostrich cannot fly) (ostrich is-a bird)
    (bird can fly) (bird has wings) (bird has feathers)
    (bird is-a animal) (fish is-a animal)
    (animal can breathe) (animal can move) (animal has skin)))
```



7

Semantic net search

```
;;; net.scm

(define (lookup object property value NETWORK)

  (define (get-parents object NET)
    (cond ((null? NET) '())
          ((and (equal? object (caar NET)) (equal? 'is-a (cadar NET)))
           (cons (caddar NET) (get-parents object (cdr NET))))
          (else (get-parents object (cdr NET)))))

  (define (inherit parents)
    (if (null? parents)
        #f
        (or (lookup (car parents) property value NETWORK)
            (inherit (cdr parents)))))

  (if (member (list object property value) NETWORK)
      #t
     (inherit (get-parents object NETWORK))))
```

```
> (lookup 'canary 'is 'yellow ANIMAL-NET)
#t

> (lookup 'canary 'can 'fly ANIMAL-NET)
#t

> (lookup 'canary 'can 'breathe ANIMAL-NET)
#t

> (lookup 'canary 'is 'green ANIMAL-NET)
#f
```

to lookup a relation

- if arc with desired label exists, done (SUCCEED)
- otherwise, if is_a relation holds, follow the link and recurse on that object/concept

8

Conceptual dependency theory

not surprisingly, early semantic nets did not scale well

- most links were general associations
- no real basis for structuring semantic relations

much research has been done in defining richer sets of links

- rely on richer formalism, not richer domain knowledge

Conceptual Dependency Theory (Schank, 1973)

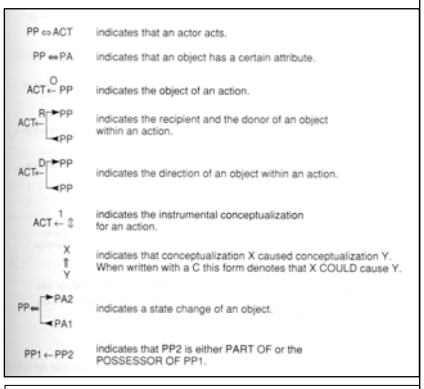
- attempts to model the semantic structure of natural language
- 4 primitive conceptualizations, from which meaning is built

ACT action
PP objects (picture producers)
AA modifiers of actions (action aiders)
PA modifiers of objects (picture aiders)

primitive actions include: ATRANS (transfer a relationship, e.g., give)
PTRANS (transfer physical location, e.g., move)
MTRANS (transfer mental information, e.g., tell)
. . .

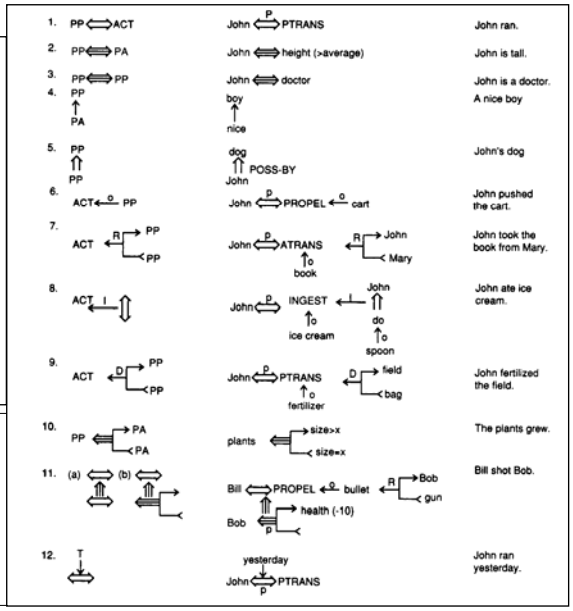
9

conceptual dependency relationships



tense/mode modifiers

p	past
f	future
t	transition
?	interrogative
/	negative
. . .	



Frames (Minsky, 1975)

in contrast to distributed knowledge networks, can instead organize knowledge into units representing situations or objects

When one encounters a new situation (or makes a substantial change in one's view of a problem) one selects from a memory structure called a "frame." This is a remembered framework to be adapted to fit reality by changing details as necessary.

-- Marvin Minsky

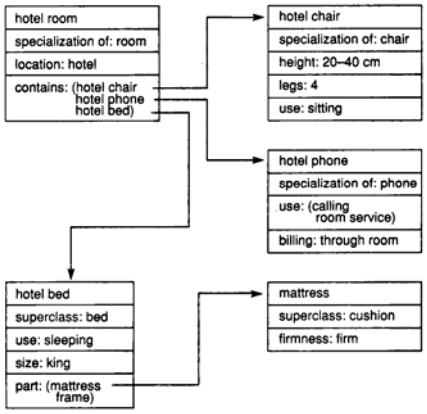
HOTEL ROOM

11

Frame example

a frame is a structured collection of data

- has *slots* (properties) and *fillers* (values)
- fillers can be links to other frames



similar to semantic nets, can perform hierarchical lookup

- if desired slot exists, get filler
- if not, follow link to parent frame and recursively look there

the structured nature of frames makes them easier to extend

- can include default values for slots
- can specify constraints on slots
- can attach procedures to slots

12

Scripts (Schank & Abelson, 1975)

a script is a structure that describes a stereotyped sequence of events in a particular context

- closely resembles a frame, but with additional information about the expected sequence of events and the goals/motivations of the actors involved
- the elements of the script are represented using Conceptual Dependency relationships (as such, actions are reduced to conceptual primitives)

EXAMPLE: restaurant script

describes: items usually found in a restaurant
people and their roles (e.g., chef, waiter, …)
preconditions and postconditions
common scenes in a restaurant: entering, ordering, eating, leaving

13

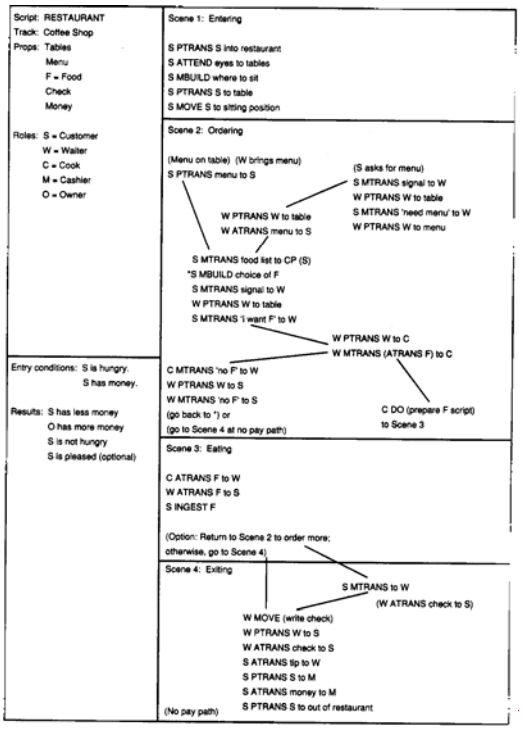
Hotel script

props and roles are identified

pre- and post-conditions

CDs describe actions that occur in each of the individual scenes

SAM: Script Applier Mechanism
- Cullingford & Schank, 1975
- was able to read simple newspaper articles, store them in CD form, summarize and answer questions



Representation applications

semantic nets, frames, and scripts were foundational structures

- more recently, they have been adapted and incorporated into hybrid structures

vision

- Minksy saw frames as representing different perspective of an object

understanding

- use frames with defaults to "fill in the blanks" in understanding
  EXAMPLE: "I looked in the janitor's closet ..."
- Lenat's AM represented concepts as frames (new concepts spawned new frames)

smart databases

- Lenat's CYC project used extension of frames, with conventions for slots & fillers
- PARKA project at Maryland uses frame-based language for efficiently accessing large knowledge bases
- Hyundai Engineering uses frame-based system for planning construction projects

interesting note:

- MIT research on frames (and similar research at XEROX PARC) led to object-oriented programming and the OOP approach to software engineering

15

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.

16

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

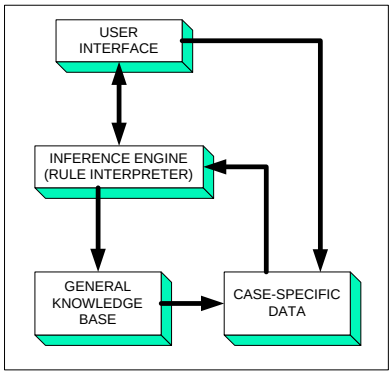
17

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.
```



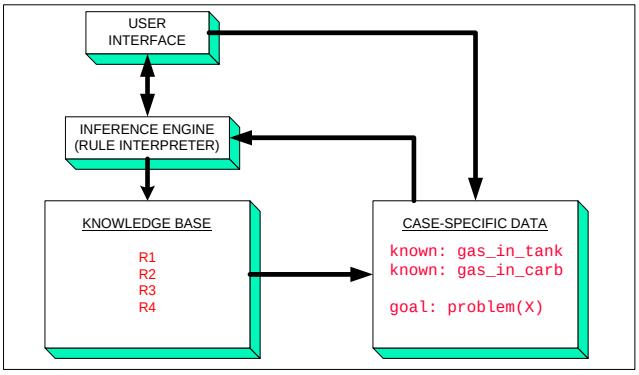
user interface	acquires information and displays results
inference engine	performs deductions on the known facts & rules (i.e., applies the knowledge base)
knowledge base	domain specific facts & rules for solving problems in the domain
case-specific data	working memory, stores info about current deduction

18

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.
```



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

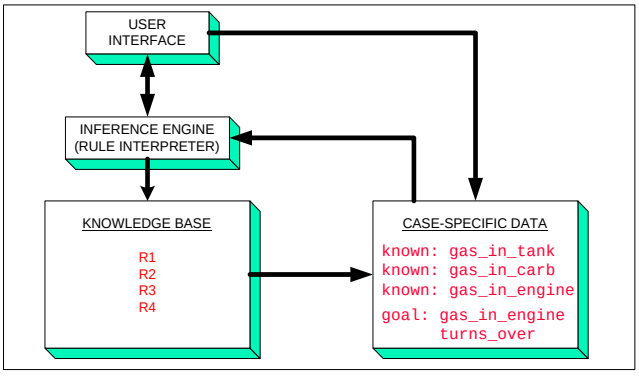
User Interface may be used to load initial facts about the specific task, specify a goal

19

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.
```



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

20

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

21

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.

22

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 $\vee$ P2) = max(CF(P1), CF(P2))
CF(P1 $\wedge$ P2) = min(CF(P1), CF(P2))

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

23

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

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 $\wedge$ sharpTeeth) * CF(R2)
= 0.3 * 0.7
= 0.21

24

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 ∧ 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)

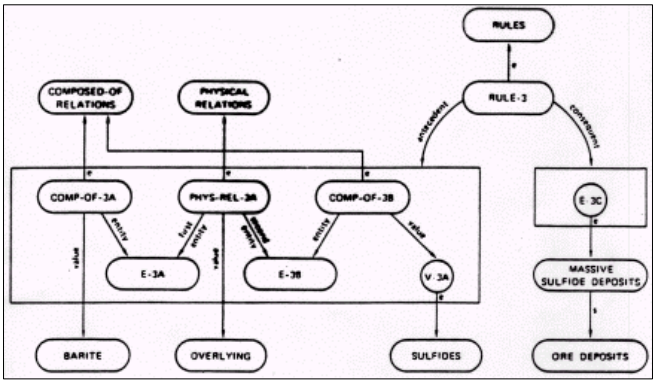
25

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.

26

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

27

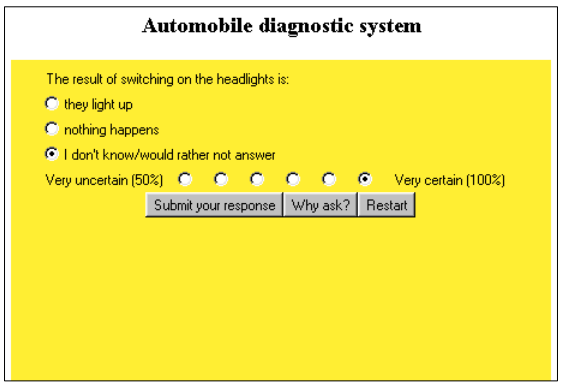
Expert system shell

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, …



view page in browser

28

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

29

… 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

30

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)

31