

Expert Systems

See [Bratko 1990]

Motivation:

- problems in specific domain areas
- solutions need specialized knowledge
- specialized knowledge is rare and expensive

Question: is automatization possible?

Endeavour: Expert Systems (XPS)

Challenges for XPS

- automated *knowledge* (not just data) processing
- specialized domain knowledge
- emulating a human expert
- explanation of decisions and reasoning
- dealing with uncertain and incomplete information

1

Narrowing the Specifications

Requirements for XPS:

- emulating human experts
- narrow domain

Applications:

- medical diagnosis
- diagnosing and locating failures in equipment
- interpreting measurement data

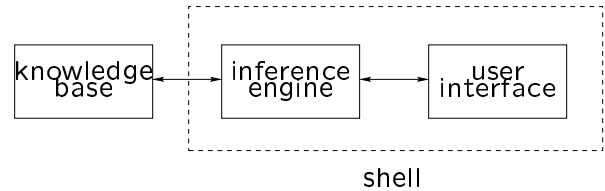
Properties of XPS

- possession and processing of knowledge
- explanation of its behaviour and decisions
- transparency (counter-check by user)
- handling uncertainty
- natural interaction
- handling incompleteness and lack of reliability

Essentials

- problem-solving capabilities
- dealing with uncertainties
- user-interaction
- explanation

Structure of an XPS



4

5

Structure of an XPS II

Knowledge Base:

- comprising the knowledge specific to domain of applications
 - simple facts
 - rules describing relations or phenomena
 - methods and heuristics for solving problems

Inference Engine: active use of the knowledge base

User Interface:

- communication between user and system
- insight into problem-solving process

Structure of an XPS III

Expert System Shell: inference engine and user interface

Design:

- XPS shell is ideally domain-independent
- new knowledge base can be plugged for each application
- caveat: in general, this will not work as smoothly as in theory

6

7

Basic Techniques

- *if-then* rules
- basic inference mechanisms
 - forward chaining
 - backward chaining
- representing uncertainty
- semantic networks
- frame-based representation of knowledge

8

Alternatively: *productions*

Examples:

- if precondition P then conclusion C
- if situation S then action A
- if conditions $C1$ and $C2$ hold then condition C does not

Features:

- limited modularity
- incrementability
- modifiability (follows from modularity)
- transparency

Following Questions Possible:

- *how* did you reach this conclusion?
- *why* did you reach this conclusion?

9

Properties of If-Then Rules

- define logical relations between concepts of problem domain
- logical relations are *categorical knowledge*, always true
- however: in most relevant domains (medical diagnosis), “soft” knowledge prevails
- empirical regularities are only valid to a certain degree
- likelihood quantification necessary

if condition A then conclusion B follows with certainty F

10

Example: MYCIN

Medical Consultation

If

1. the infection is primary bacteremia, and
2. the site of the culture is one of the sterilesites, and
3. the suspected portal of entry of the organism is the gastrointestinal tract

Then

there is suggestive evidence (0.7) that the identity of the organism is bacteroides.

11

If

the pressure in V-01 reached relief valve lift pressure

Then

the relief valve on V-01 has lifted
[$N = 0.005$, $S = 400$]

If

NOT the pressure in V-01 reached relief valve lift pressure, and the relief valve on V-01 has lifted

Then

the V-01 relief valve opened early (the set pressure has drifted) [$N = 0.001$, $S = 2000$]

Rule Validity

Necessity: N – how necessary the condition is for the conclusion to be true

Sufficiency: S – how sufficient the condition is for the conclusion to be true

Knowledge “Engineering”

Task: developing a larger XPS

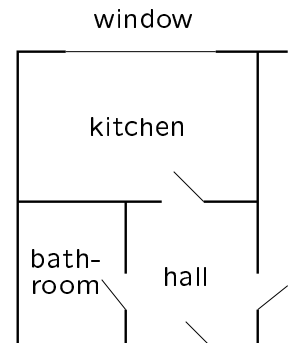
Problem:

- actual expert knowledge necessary
- developer of expert system is usually not domain expert
- domain experts have to be consulted
- implicit knowledge has to be made explicit and be organized

Knowledge Elicitation:

- special interview techniques necessary
- expert's knowledge strongly internalized
- expert's thinking not necessarily numeric or rule-based
- significant effort to formalize that knowledge

Example: Water Leaking



Problem: water leak

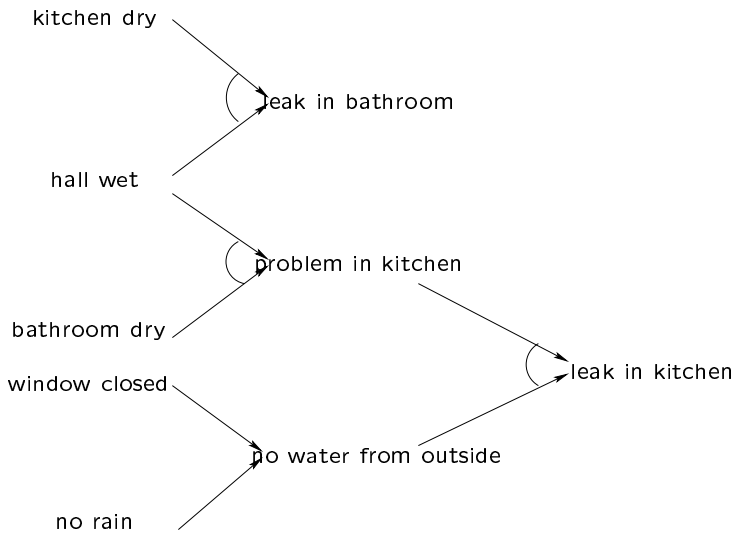
Assumptions:

- leakage causes problem in hall
- only single faults assumed (bathroom or kitchen)

Approach:

- inference network
- nodes: propositions
- links: rules
- arcs: conjunctive connections between propositions

Inference Network



- AND/OR graph

16

Forward/Backward Chaining

Condition: if/then rules

Methods: searching AND/OR graphs

Requirements: similarity to human train of thought

17

Backward Chaining

1. start with hypothesis `leak_in_kitchen`
2. need to confirm hypothesis
3. `problem_in_kitchen` and `no_water_from_outside` must be true for hypothesis to hold
4. confirmed, e.g. if hall is wet and bathroom is dry
5. modeled by Prolog in natural way

18

Prolog Program

```
leak_in_bathroom :-  
    hall_wet,  
    kitchen_dry.  
  
problem_in_kitchen :-  
    hall_wet,  
    bathroom_dry.  
  
no_water_from_outside :-  
    window_closed  
    ;  
    no_rain.  
  
leak_in_kitchen :-  
    ...
```

19

Program Kitchen

20

Disadvantages of the Approach

- user has to state facts in advance
- which facts are necessary?
- expert system shell

21

Forward Chaining

Backward Chaining

- starting with hypothesis
- working backwards
- towards findings and facts

Forward Chaining:

- starts from facts
- conclusions
- programming easy, but not trivial in prolog

22

Discussion

Backward Chaining: goal driven

Forward Chaining: data driven

Real Life: using mixture of forward and backward chaining

23

Explanations

Question “How”: how was an answer derived.
Example: leak found in the kitchen *because*

- there is a problem in the kitchen, which was concluded from hall wet and bathroom dry and
- no water came from outside, which was concluded from window closed

24

Proof Tree: representation is

1. If P is a fact then the proof tree is P.
2. If P was derived using a rule
if Cond then P
then the proof tree is
P \Leftarrow ProofCond
where ProofCond is a proof tree of Cond.
3. Let P1 and P2 be propositions with proof trees Proof1 and Proof2. If P is P1 and P2 then proof tree is Proof1 and Proof2. If P is P1 or P2 then a proof tree is given by Proof1 or Proof2.

Recapitulate: proofs for predicate logic

Note: proof trees are essentially solution trees for AND/OR graph problems

Question “Why”: needed interactively by user: “Why are you asking this question?”

25

Uncertainty

Qualification of Facts: true, highly likely, likely, unlikely, impossible

Quantification by Numbers: e.g. numbers between 0 and 1

Form:

Proposition : CertaintyFactor

and for rules

if Condition then Conclusion : Certainty

26

Uncertainty II

Combination: of rule certainties

$$c(P1 \text{ and } P2) = \min(c(P1), c(P2))$$

$$c(P1 \text{ or } P2) = \max(c(P1), c(P2))$$

Rule Certainty: for a rule

if P1 then P2 : C

the certainty propagates as

$$c(P2) = c(P1 \cdot C)$$

27

Certainties: given by

Problems

```
given(Proposition, Certainty).
```

Soften Rules:

```
if hall_wet and bathroom_dry
then problem_in_kitchen : 0.9
```

Rule Interpreter:

```
certainty(P, Cert) :-
    given(P, Cert).

certainty(Cond1 and Cond2, Cert) :-
    certainty(Cond1, Cert1),
    certainty(Cond2, Cert2),
    minimum(Cert1, Cert2, Cert).

certainty(Cond1 or Cond2, Cert) :-
    certainty(Cond1, Cert1),
    certainty(Cond2, Cert2),
    maximum(Cert1, Cert2, Cert).

certainty(P, Cert) :-
    if Cond then P : C1,
    certainty(Cond, C2),
    Cert is C1 * C2.
```

28

Consider:

- certainty of a is 0.5
- certainty of b is 0

Then: certainty of a or b is 0.5.

What if:

- certainty of a is 0.5
- certainty of b is 0.5

Still: certainty of a or b is 0.5.

29

Problem of Combinations for Proposition Certainties

Different Approaches:

- Lukasiewicz Norms ('and' and 'or' versions)
- Constructions

Further Problems:

- if a or b then c
- certainty of c does depend not only on a and b individually, but on correlation between a and b
- where to get the correlation information?
- independence assumed
- this is not generally true
- handling of uncertainties
- human use unsafe assumptions

30

Handling Uncertainties

Solution Approaches:

- formalism to handle uncertainties:
Fuzzy Logic
- uncertainty picture of theory of probability:
Bayesian View

31

Semantic Networks

Semantic Networks and Frames

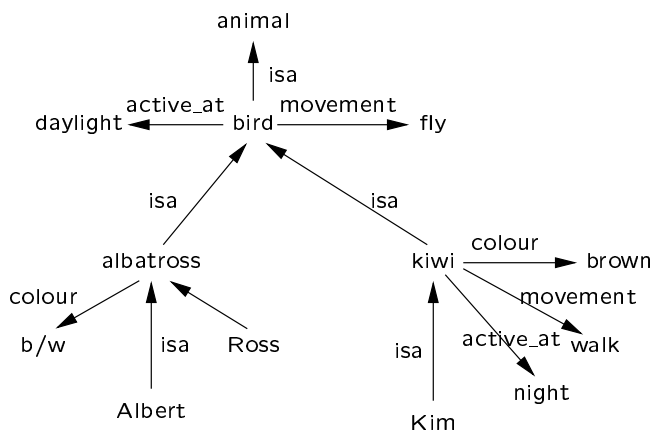
- alternatives for representing knowledge
- adding structure to rule-based representations
- facts can be abstracted away

- graph
- entity/relationship
- example:
 - a bird is a kind of animal
 - birds usually fly
 - an albatross is a bird
 - albert and ross is an albatross

32

33

Semantic Networks: Example



Semantic Networks in Prolog: Demo

34

35

Frames:

- facts clustered *around* objects
- frame contains slots and is related to an object
- slots store information or may be empty
- slots can be filled by inference

Example:

```
FRAME: bird
a_kind_of: animal
moving_method: fly
active_at: daylight
```

```
FRAME: albatross
a_kind_of: bird
colour: black_and_white
size: 115
```

```
FRAME: kiwi
a_kind_of: bird
moving_method: walk
active_at: night
colour: brown
size: 40
```

```
FRAME: Albert
instance_of: albatross
size: 120
```