CS370

Symbolic Programming

Declarative Programming

LECTURE 15: Knowledge Representation and Expert Systems

Jong C. Park

park@cs.kaist.ac.kr

Computer Science Department

Korea Advanced Institute of Science and Technology

http://nlp.kaist.ac.kr/~cs370

Knowledge Representation and Expert Systems

- Functions and structure of an expert system
- Representing knowledge with *if-then* rules
- Forward and backward chaining in rule-based systems
- Generating explanation
- Introducing uncertainty
- Semantic networks and frames

Functions and structure of an expert system

Typical applications of an expert system

- medical diagnosis
- locating a failure in an equipment
- interpreting measurement data

Functions and structure of an expert system

Issues

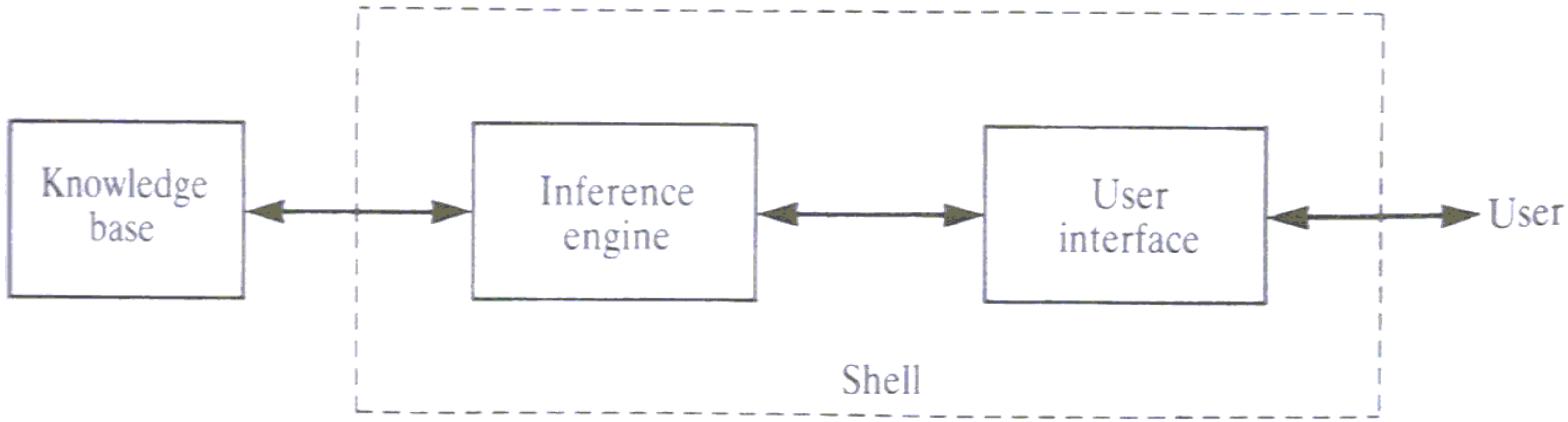
- expert knowledge
- explanation
- uncertainty and incompleteness of the knowledge

Functions of an expert system

- problem-solving function
- user-interaction function

Functions and structure of an expert system

⊙ Modules of an expert system



⊙ Why separate knowledge from algorithms?

-

Representing knowledge with *if-then* rules

Examples of production rules

- if precondition P then conclusion C

  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.

Representing knowledge with *if-then* rules

⦿Examples of production rules

- if situation S then action A

  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 conditions C1 and C2 hold then condition C does not hold

Representing knowledge with *if-then* rules

Why production rules?

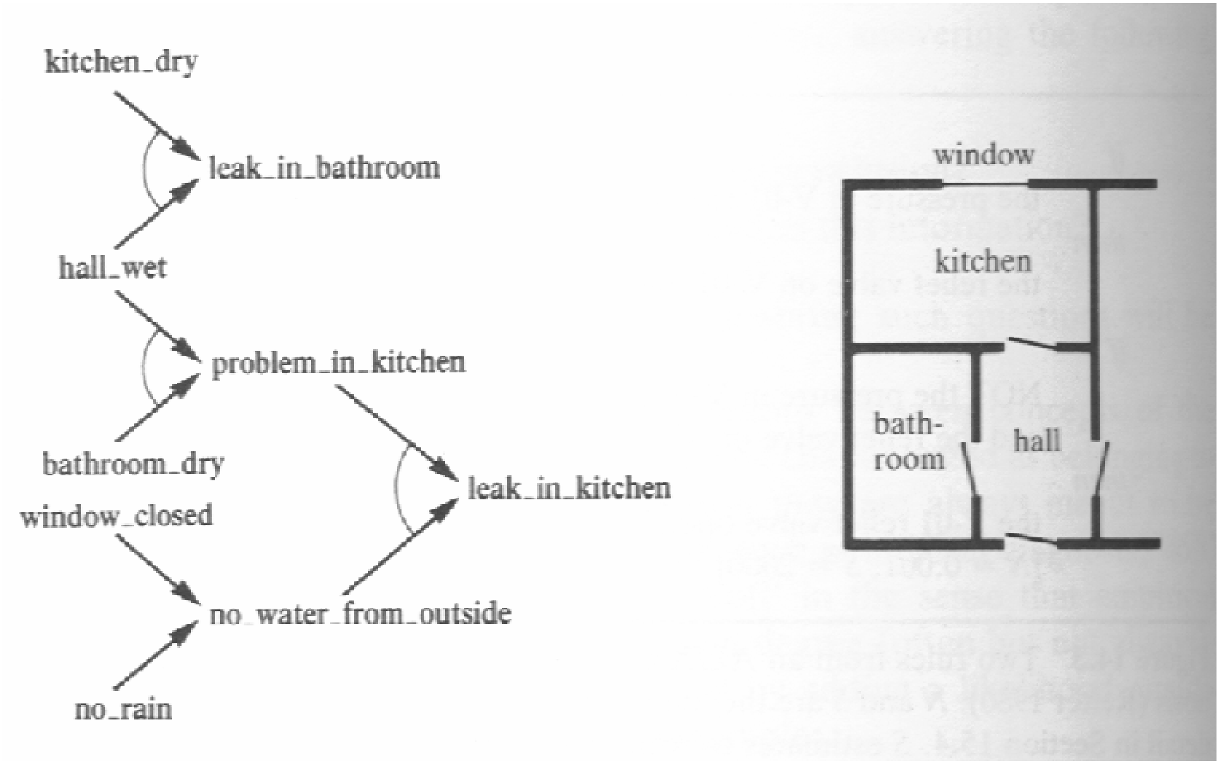
- modularity
- incrementality
- modifiability
- transparency

Questions

- How
  - How did you reach this conclusion?
- Why
  - Why are you interested in this info.?

Representing knowledge with *if-then* rules

⦿ A toy knowledge base



Forward and backward chaining

Backward chaining

- from the hypothesis to the pieces of evidence
- use Prolog's own syntax for rules

```
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 :- problem_in_kitchen,
                   no_water_from_outside.
hall_wet.     bathroom_dry.     window_closed.
?- leak_in_kitchen.
```

Forward and backward chaining

⊙ Backward chaining

- define new operators for if, then, or, and and.

```
:- op(800,fx,if).
:- op(700, xfx, then).
:- op(300,xfy,or).
:- op(200,xfy,and).
if hall_wet and kitchen_dry
then leak_in_bathroom.
```

- observable findings as a procedure fact

```
fact(hall_wet).
fact(bathroom_dry).
```

Forward and backward chaining

Backward chaining (Figure 15.6)

- new interpreter: is_true(P)

```
is_true(P) :- fact(P).
is_true(P) :- if Condition then P,
              is_true(Condition).
is_true(P1 and P2) :- is_true(P1), is_true(P2).
is_true(P1 or P2) :- is_true(P1) ; is_true(P2).
```

Forward and backward chaining

Forward chaining

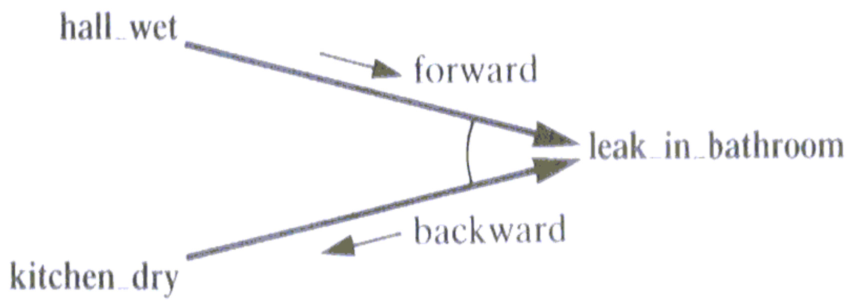
- from confirmed findings to the final conclusion
  - if Condition then Conclusion

```
forward :-  new_drived_fact(P),  !,
          write('Derived: '), write(P), nl, assert(fact(P)),
          forward ; write('No more facts').
new_derived_fact(Concl) :-
          if Cond then Concl, not fact(Concl),
          composed_fact(Cond).
composed_fact(Cond) :- fact(Cond).
composed_fact(Cond1 and Cond2) :-
         composed_fact(Cond1), composed_fact(Cond2).
composed_fact(Cond1 or Cond2) :-
         composed_fact(Cond1) ; composed_fact(Cond2).
```

Forward and backward chaining

⦿Comparison

- goal-driven vs. data-driven
  - data → … → goals
  - evidence → … → hypotheses
  - findings, observations → … → explanations
- What are the measures?
  - data nodes vs. goal nodes
- hybrid



Generating explanation

⦿Explaining how and why

- how
  - give the proof tree of the final conclusion
  - modify the predicate is_true (Figure 15.8)

```
is_true(P,P) :- fact(P).
is_true(P,P<=CondProof) :-
  if Cond then P,
  is_true(Cond,CondProof).
is_true(P1 and P2,Proof1 and Proof2) :-
  is_true(P1,Proof1), is_true(P2,Proof2).
```

  - cf. the solution trees in AND/OR graphs

Generating explanation

Explaining how and why

- why
  - required during the reasoning process
  - requires user interaction with the reasoning process
  - Chapter 16

Introducing uncertainty

- **Categorical answers and implications**
  - either true or false, not somewhere between
- **Qualified answers and implications**
  - *true, highly likely, likely, unlikely, impossible*
  - the degree of belief, certainty factor, measure of belief, subjective certainty
  - Figure 15.9: An interpreter for rules with certainties

Introducing uncertainty

⦿Combining the certainties

- Proposition : CertaintyFactor
- if Condition then Conclusion : Certainty
- c(P1 and P2) = min(c(P1),c(P2))
- c(P1 or P2) = max(c(P1),c(P2))
- c(P2) = c(P1)*C ← if P1 then P2 : C

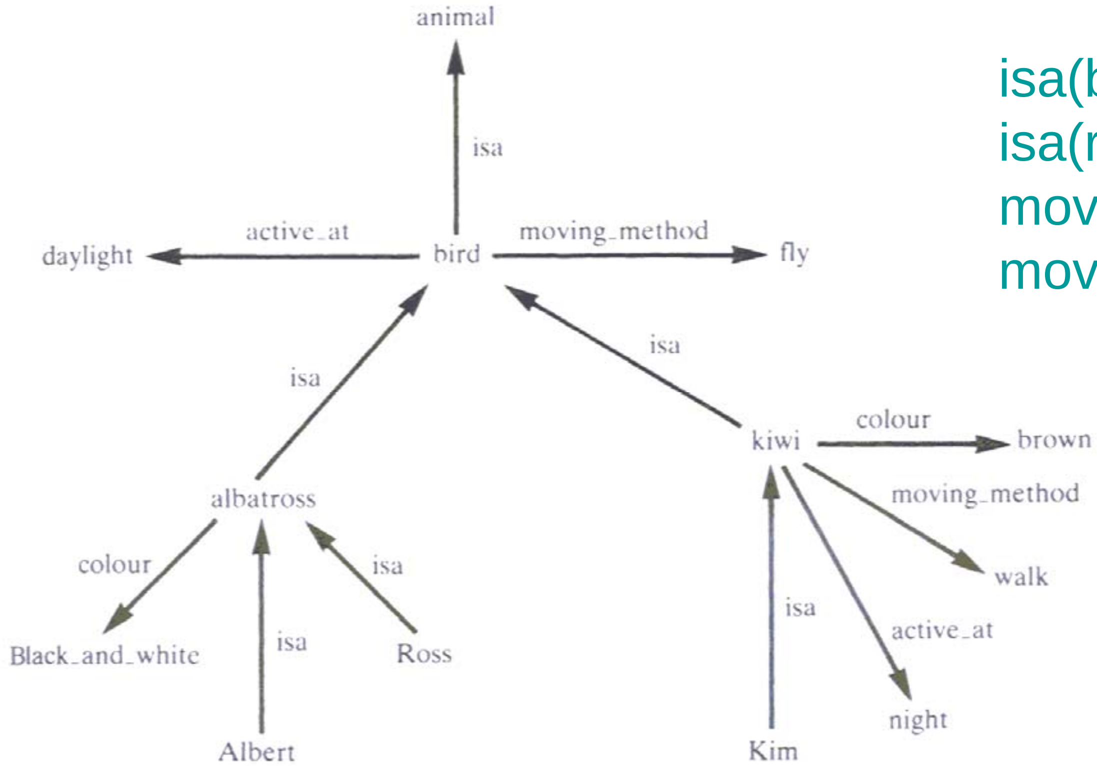
```
?- certainty(leak_in_kitchen,C).
C = 0.8
```

Introducing uncertainty

- **Controversial issues**
  - the usefulness of a probability theory
  - drawbacks of *ad hoc* uncertainty schemes
- **Modeling dependencies**
  - mathematical soundness
  - realistic correctness

Semantic networks and frames

⦿ Semantic networks: entities and relations



isa(bird,animal).
isa(ross,albatross).
moving_method(bird,fly).
moving_method(kiwi,walk).

Inheritance

- the method of moving

```
moving_method(X,Method) :-
  isa(X,SuperX),
  moving_method(SuperX,Method).
```

- more general rule about facts

```
fact(Fact) :- Fact, !.
fact(Fact) :- Fact =.. [Rel,Arg1,Arg2],
  isa(Arg1,SuperArg),
  SuperFact =.. [Rel,SuperArg,Arg2],
  fact(SuperFact).
?- fact(moving_method(kim,Method)).
Method = walk
```

Semantic networks and frames

⦿ Frames: facts about objects (Figure 15.14)

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

FRAME: albatross
a_kind_of: bird
color: black_and_white
size: 115

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

FRAME: Albert
a_kind_of: albatross
size: 120

Semantic networks and frames

⦿Frames in Prolog & retrieving facts

- Frame_name(Slot,Value)
- value(Frame,Slot,Value)

```
value(Frame,Slot,Value) :-
  Query =.. [Frame,Slot,Value], call(Query), !.
value(Frame,Slot,Value) :-
  parent(Frame,ParentFrame),
  value(ParentFrame,Slot,Value).
parent(Frame,ParentFrame) :-
  (Query =.. [Frame,a_kind_of,ParentFrame];
   Query =.. [Frame,instance_of,ParentFrame]), call(Query).
?- value(albert,active_at,AlbertTime).
AlbertTime = daylight
```

Summary

- Functions and structure of an expert system
- Representing knowledge with *if-then* rules
- Forward and backward chaining in rule-based systems
- Generating explanation
- Introducing uncertainty
- Semantic networks and frames