ARTIFICIAL INTELLIGENCE

KNOWLEDGE REPRESENTATION USING OTHER LOGIC

Ch6: Knowledge Representation Using Rules

- Procedural vs. Declarative Knowledge
- Logic Programming
- Forward vs. backward reasoning
- Matching
- Control knowledge

Procedural vs. Declarative Knowledge

Declarative representation

- Knowledge is specified but the use is not given.

- Need a program that specifies what is to be done to the knowledge and how.

- Example:

    • Logical assertions and Resolution theorem prover

- A different way: Logical assertions can be viewed as a program, rather than as data to a program.

=> Logical assertions = Procedural representations of knowledge

Procedural vs. Declarative Knowledge

Procedural representation

- The control information that is necessary to use the knowledge is considered to be embedded in the knowledge itself.

- Need an interpreter that follows the instructions given in the knowledge.

- The real difference between the declarative and the procedural views of knowledge lines in where control information resides.

  - Kowalski's equation: Algorithm = Logic + Control

Procedural vs. Declarative Knowledge

man(Marcus)

man(Caesar)

$\forall x$: man(x) $\rightarrow$ person(x)

person(Cleopatra)

person(x)?

Logic Programming

Logical assertions are viewed as programs

Most popular programming system: PROLOG

Prolog program = {Horn Clauses}

- Horn clause: disjunction of literals of which at most one is positive literal

$$\neg P_1 \lor \neg P_2 \lor \ldots \lor \neg P_n \lor Q$$

$$P_1 \land P_2 \land \ldots \land P_n \Rightarrow Q$$

=> Prolog program is decidable

- Control structure: Prolog interpreter = backward reasoning + depth-first with backtracking

Logic Programming

Logic:

$\forall X$: pet(X) ^ small(X) → apartmentpet(X)

$\forall X$: cat(X) v dog(X) → pet(X)

$\forall X$: poodle(X) → dog(X) ^ small(X)

poodle(fluffy)

Prolog:

```
apartmentpet(X) :- pet(X) , small(X).
pet(X) :- cat(X).
pet(X) :- dog(X).
dog(X) :- poodle(X).
small(X) :- poodle(X).                poodle(fluffy).
```

Logic Programming

Prolog vs. Logic

- Quantification is provided implicitly by the way the variables are interpreted.
  - Variables: begin with UPPERCASE letter
  - Constants: begin with lowercase letters or number
- There is an explicit symbol for AND (,), but there's none for OR. Instead, disjunction must be represented as a list of alternative statements
- "p implies q" is written as q :- p.

Logic Programming

Prolog: How to find a solution?

?- apartmentpet(X)

Logical negation ¬ cannot be represented explicitly in pure Prolog.

- Example: $\forall x$: dog(x) $\rightarrow$ ¬cat(x)

=> problem-solving strategy: NEGATION AS FAILURE

?- cat(fluffy). => false b/c it's unable to prove Fluffy is a cat.

Negation as failure requires: CLOSED WORLD ASSUMPTION

- True assertions are contained in our knowledge base or derivable from assertions that are so contained

Logic Programming

The occur-check is omitted from the unification: unsound

```
test:- p(X, X).
 p(Y, f(Y)).
 ?- test  => yes
```

Backward chaining with depth-first search: incomplete

```
p(X, X).
p(X, Y):- q(X, Y).
q(X, Y):- q(Y, X).
?- p(2, 2) => yes
```

```
p(X, Y):- q(X, Y).
p(X, X).
q(X, Y):- q(Y, X).
?- p(2, 2) => loop
```

Unsafe cut: incomplete

Forward vs. Backward Reasoning

Forward: from the start states.

Backward: from the goal states.

Forward or Backward?

=> The topology of the problem space

Forward vs. Backward Reasoning

Forward or Backward?

- Move from the smaller set of states to the larger set of states

- Proceed in the direction with the lower branching factor

- Proceed in the direction that corresponds more closely with the way the user will think

- Proceed in the direction that corresponds more closely with the way the problem-solving episodes will be triggered

Forward vs. Backward Reasoning

Bidirectional search (hybrid reasoning)

- Search both forward from the start state and backward from the goal simultaneously until two paths meet somewhere in between.

- Combining Forward and Backward Reasoning

$$\leftarrow A_1, \ldots, A_{k-1}, A_k, A_{k+1}, \ldots, A_n$$

achieved by forward reasoning

achieved by backward reasoning

Forward vs. Backward Reasoning

To encode the knowledge for reasoning, we need 2 kinds of rules:

- Forward rules: to encode knowledge about how to respond to certain input.

- Backward rules: to encode knowledge about how to achieve particular goals.

Matching

How to extract from the entire collection of rules that can be applied at a given point?

=> Matching between current state and the precondition of the rules

Matching

Indexing

- A large number of rules => too slow to find a rule
- Indexing: Use the current state as an index into rules and select the matching ones immediately
- Only works when preconditions of rules match exact board configuration
- It's not always obvious whether a rule's preconditions are satisfied by a particular state.
- There's a trade-off between the ease of writing rules (high-level descriptions) and the simplicity of the matching process

Matching

Matching with variables

- Generality in the statements of the rules

⇒ Need a search process to discover a match between a particular state and the preconditions of a given rule

- Backward-chaining systems:

  - One-one matching algorithm:

    →Unification procedure + Depth-first backtracking to select individual rules

- Forward-chaining systems:

  - Many-many matching algorithm: RETE

Matching

RETE network (Forgy, 1982)

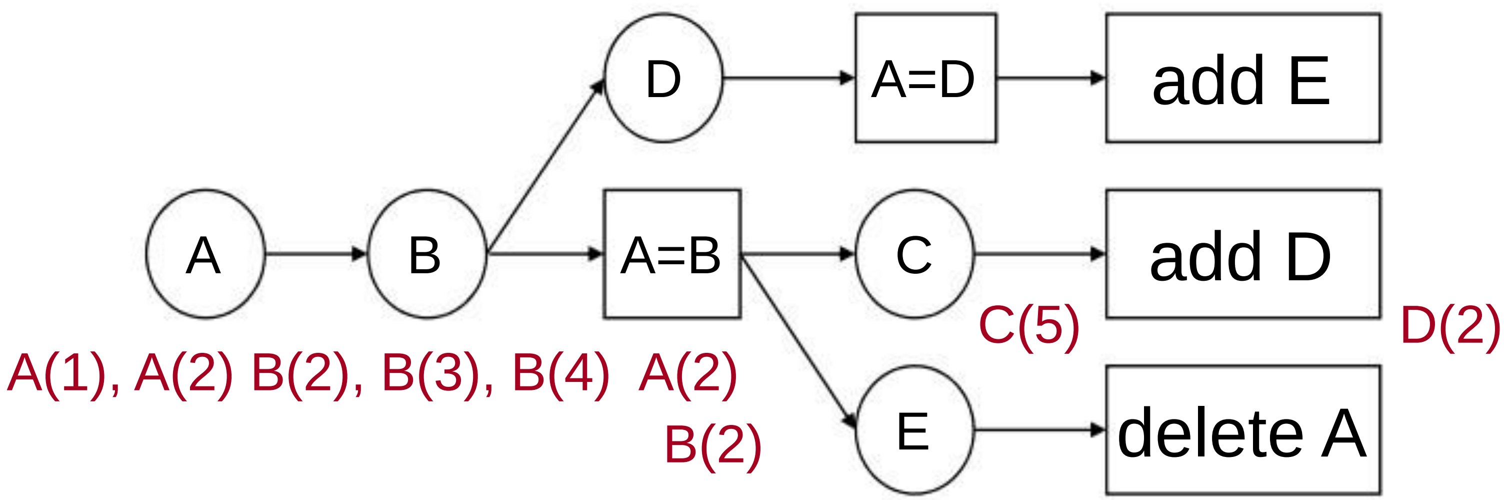
Working memory: {A(1), A(2), B(2), B(3), B(4), C(5)}

Rule memory:

$A(x) \wedge B(x) \wedge C(y) \Rightarrow$ add $D(x)$

$A(x) \wedge B(y) \wedge D(x) \Rightarrow$ add $E(x)$

$A(x) \wedge B(x) \wedge E(y) \Rightarrow$ delete $A(x)$



Matching

Approximate matching

- Rules should be applied if their preconditions approximately match the current situation

- Example: A speech-understanding program

  • Rules: A description of a physical waveform to phones (a, e, …)

  • Physical signal: differences in the way individuals speak, result of background noise, …

Matching

ELIZA (Weizenbaum, 1966):

Person: It's true. I am unhappy.

ELIZA: Do you think coming here will help you not to be unhappy?

Person: I need some help, that much seems certain.

ELIZA: What would it mean to you if you got some help?

Person: Perhaps I could learn to get along with my brother.

ELIZA: Tell me more about your family.

Matching

ELIZA (Weizenbaum, 1966):

(X me Y) → (X you Y)

(I remember X) → (Why do you remember X just now?)

(My {family-member} is Y) → (Who else in your family is Y?)

(X {family-member} Y) → (Tell me more about your family)

Person: My brother is mean to me.

ELIZA: Who else in your family is mean to you?

Matching

Conflict resolution:

- Preferences based on rules:
  - Specificity of rules
  - Physical order of rules
- Preferences based on objects:
  - Importance of objects
  - Position of objects
- Preferences based on states:
  - Evaluation of states

Control Knowledge

Search control knowledge: Knowledge about which paths are most likely to lead quickly to a goal state.

- Which states are more preferable to others.
- Which rule to apply in a given situation.
- The order in which to pursue subgoals
- Useful sequences of rules to apply.

Search control knowledge = Meta knowledge

SUMMARY

- LOGIC
- MATCHING
- CONFLICTS