Artificial Intelligence
Chapter 17
Knowledge-Based Systems

Biointelligence Lab
School of Computer Sci. & Eng.
Seoul National University

Outline

- Confronting the Real World
- Reasoning Using Horn Clauses
- Maintenance in Dynamic Knowledge Bases
- Rule-Based Expert Systems
- Rule Learning
- Additional Readings and Discussion

17.1 Confronting the Real World (1/3)

- Knowledge-based systems
  - Programs that reason over extensive knowledge bases.
  - Do the methods scale up sufficiently well for practical applications?
- Three major theoretical properties of logical reasoning systems
  - Soundness
    - To be confident that an inferred conclusion is true
  - Completeness
    - To be confident that inference will eventually produce any true conclusion
  - Tractability
    - To be confident that inference is feasible

17.1 Confronting the Real World (2/3)

- Predicate calculus
  - Resolution refutation is sound and complete, but semi-decidable.
    - Semi-decidablility makes the predicate calculus inherently intractable.
    - Even on problems for which resolution refutation terminates, the procedure is NP-hard.
    - This fact has led many to despair of using formal, logical methods for large-scale reasoning problems.

17.1 Confronting the Real World (3/3)

- To make reasoning more efficient
  - We could use procedures that might occasionally prove an untrue formula.
  - We could use procedures that might not be guaranteed to find proofs of true formulas.
  - We could use a language that is less expressive than the full predicate calculus.
    - Reasoning is typically more efficient with Horn clauses.

17.2 Reasoning Using Horn Clauses

- Horn Clauses
  - Clauses that have at most one positive literal.
  - Rule: $\lambda_h :- \lambda_{b_1}, ..., \lambda_{b_n}$
    - There is at least one negative literal and a single positive literal.
    - The literal λ_h is called the head of the clause.
    - The ordered list of literals, $\lambda_{b1}, \ldots, \lambda_{bn}$, is called body.
  - Fact: $\lambda_h :-$
    - There may be no negative literals in the clause.
  - Goal: $:- \lambda_{b_1}, ..., \lambda_{b_n}$
    - There may be no positive literal in the clause.
  - Horn clauses form the basis of the programming language PROLOG.

Inference over PROLOG Clauses (1/2)

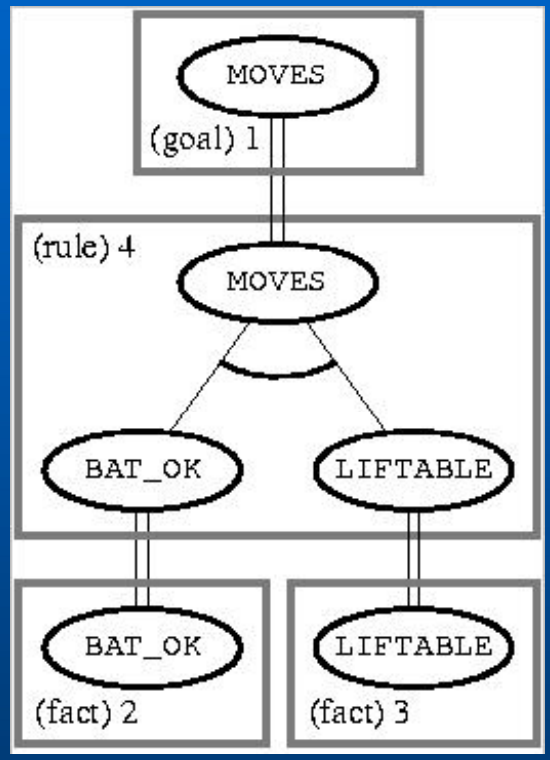
- Resolution Operation
  - A goal can resolve with a fact by unifying the fact with one of the literals in the goal.
    - The resolvent is a new goal consisting of a list of all of the substitution instances of the other literals in the original goal.
  - A goal can resolve with a rule by unifying the head of the rule with one of the literals in the goal.
    - The resolvent is a new goal formed by appending the list of substitution instances of all of the literals in the body of the rule to the front of the list of substitution instances of all of the other (nonresolved-upon) literals in the goal.

Inference over PROLOG Clauses (2/2)

- PROLOG programs
  - The clauses are usually ordered in the following way: goal, facts, rules.
  - Proof of a goal clause succeeds when the new goal produced by a resolution is empty.
  - Depth-first, backtracking search procedure
    - A goal clause fails if the interpreter has tried all resolutions for one of the goal literals and none results in new goals that can be proved.
    - The interpreter backtracks to the previous goal clause and tries other resolutions on it.

Example of an AND/OR Tree (1/2)

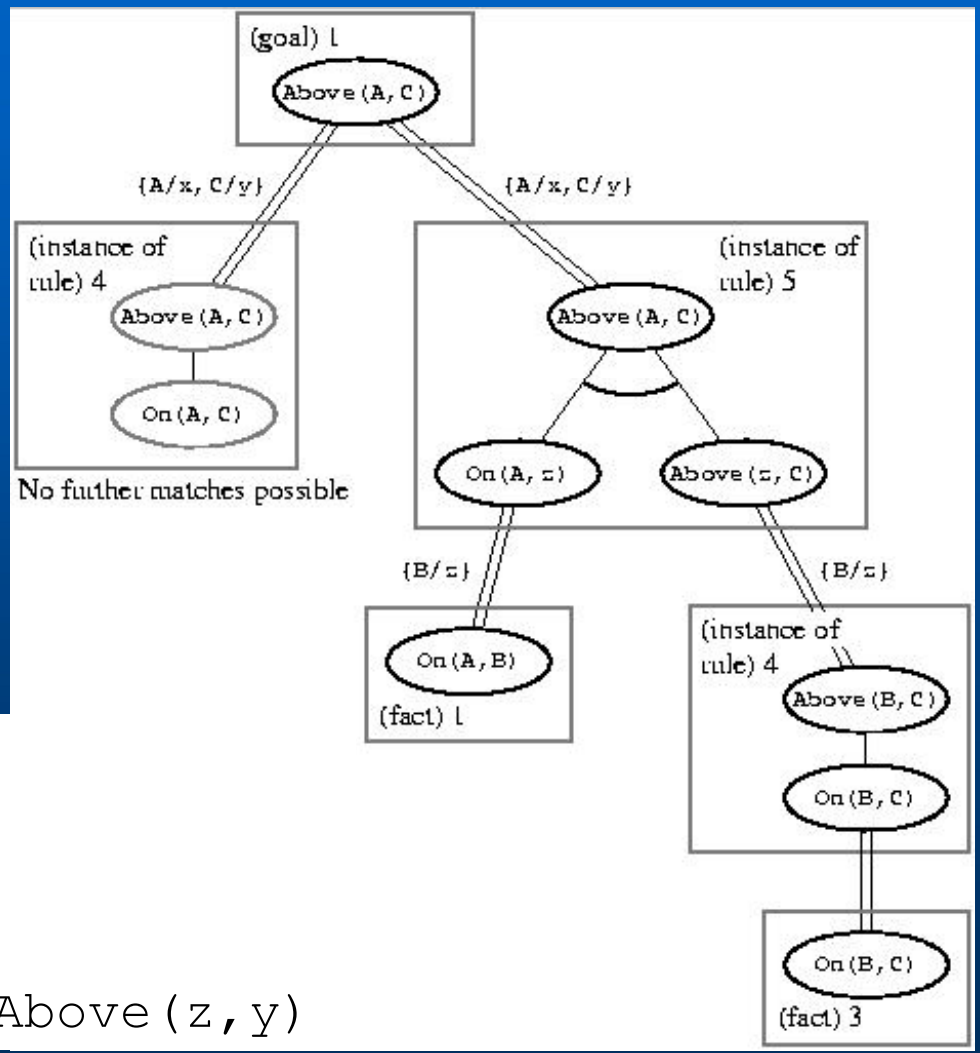
- Body nodes are called *AND* nodes because they must all be proved.
- If there had been alternative resolutions possible, the search for a proof tree could have generated additional nodes, called *OR* nodes.




```
1.  :- MOVES
2.  BAT_OK :-
3.  LIFTABLE :-
4.  MOVES :- BAT_OK, LIFTABLE
```

Example of an AND/OR Tree (2/2)

- Consistency requires that the same term be substituted for the variable throughout the proof tree.




```
1.  :- Above(A,C)
2.  On(A,B)  :-
3.  On(B,C)  :-
4.  Above(x,y)  :- On(x,y)
5.  Above(x,y)  :- On(x,z), Above(z,y)
```

Forward Chaining

- Forward chaining system (e.g., OPS5)
  - Reasoning proceeds forward, beginning with facts, chaining through rules, and finally establishing the goal.
    - A rule is applicable if each of the literals in its antecedent can be unified with a corresponding fact.
    - When more than one rule is applicable, some sort of external *conflict resolution* scheme is used to decide which rule will be applied.
  - Model of aspects of human cognitive processing
    - Facts: *working* of *short-term* memory
    - Rules: *long-term* memory

17.3 Maintenance in Dynamic Knowledge Bases (1/5)

- Knowledge base of propositional calculus atoms
  - The atoms are called *premisses* instead of facts.
  - The rules are not restricted to being Horn rules.
  - A spreadline reasoning system
    - Any cell that has a formula in it gets its vale immediately calculated from the values of components.
    - If the values of any components of a formula happen to change, the value of that formula is automatically changed.

P	Q	R	S
1	1	= P ∧ Q	= P ∨ R

P	Q	R	S
1	1	1	1

P	Q	R	S
1	0	0	1

17.3 Maintenance in Dynamic Knowledge Bases (2/5)

- Truth maintenance systems (TMSs)
  - *Dynamic knowledge base*: the spreadline is extended to contain any number of premisses and rules.
    - The formulas entered in the values of rule cells are called *justifications*.
  - The values of certain premiss cells can be omitted.
    - When a cell has a value of 1 or 0, it is called IN; otherwise, out.
    - Allowing values of cells to be OUT permits an interesting generalization of rule types (e.g, U if R is *True* and S is Out).

P	Q	R	S
		= P ∧ Q	= P ∨ R

Instances of the **KB**

P	Q	R	S
1			1

P	Q	R	S
	1		

17.3 Maintenance in Dynamic Knowledge Bases (3/5)

- TMSs should be thought of as knowledge base maintenance systems that perform automatic updates when the truth values of certain atoms are changed.
  - The use of IN's and OUT's permits an elementary sort of "monotonic" or defeasible inference not sanctioned by ordinary reasoning systems.
- Cycles among the rules present some difficulties.
  - One is *mutual justification*.

Formulas

P	Q	R
	= P ∨ R	= Q

Starting KB

P	Q	R
1	1	1

Stages in calculation when P becomes OUT:

Stage 1: calculate new Q, using its formula and previous value of R

P	Q	R
OUT	1	1

Stage 2: calculate new R, given previous values of P and Q

P	Q	R
OUT	1	1

17.3 Maintenance in Dynamic Knowledge Bases (4/5)

- Assumption-based truth maintenance systems (ATMSs)
  - The spreadline is used in a backward direction.
  - We start with a particular cell's formula and ask which premiss values will make the formula in that cell *True*.
    - Other formulas called the *background theory* are considered together.
  - *Labels* of the cells
    - The values of the cells are taken to be formulas-expressed in DNF form in terms of the premiss atoms.
  - ATMSs can be used for a variety of purposes.
    - One is to perform diagnosis.

17.3 Maintenance in Dynamic Knowledge Bases (5/5)

- Conversion of a TMS to an ATMS