Lecture 40: Logic Programming

CSC 131
Fall, 2006

History

- Alan Robinson 1965 -- Resolution Theorem Prover
- Kowalski & Colmerauer 1974 -- Prolog
- Adopted as key technology in Japanese 5th Generation Project for intelligent systems 1982-91
- More recently constraint logic programming

Declarative Programming

- Specification, no control -- relational
- Three types of statements:
  - Facts: father(albert, jeffrey).
  - Rules:
    - grandparent(X,Z) :- parent(X,Y),parent(Y,Z).
    - parent(X,M):-mother(X,M).
    - parent(X,F):-father(X,F).
  - Queries
    - ?-grandparent(X,jeffrey).

See family.pro

More examples

```
ins_sort([],[]) .

ins_sort([H|T],Sorted) :-
        ins_sort(T,Rest),
        insert(H,Rest,Sorted) .

insert(Elt,[],[Elt]) .

insert(Elt,[H|T],[Elt,H|T]) :-
        (Elt=<H) .

insert(Elt,[H|T],[H|Sorted]) :-
        insert(Elt,T,Sorted) .
```

Computation

- Depth-first search via backtracking using unification on free variables.
- Look for facts or use rules backwards until find facts verifying answer -- results in proof.
- If requested (w/";") then get multiple answers

Data structures

```
insert_tree(X,nil,make_tree(nil,X,nil)).
insert_tree(X,make_tree(L,Val,R),make_tree(L,Val,NewR)):-
        X>Val,
        insert_tree(X,R,NewR).
insert_tree(X,make_tree(L,Val,R),make_tree(NewL,Val,R)):-
        X=<Val,
        insert_tree(X,L,NewL).

/* build_tree(L,Tree) converts list L into sorted tree */

build_tree([],nil).
build_tree([H|T],Tree):-
        build_tree(T,Part_tree),
        insert_tree(H,Part_tree,Tree).
```

More Treesort

```
/* in_order(Tree,List) in-order traversal of
Tree results in List */

in_order(nil,[]).
in_order(make_tree(L,Val,R),List):-
        in_order(L,First),
        in_order(R,Second),
        append(First,[Val|Second],List).

tree_sort(List,Result):-build_tree(List,Tree),
        in_order(Tree,Result).
```

Power of Non-Deterministic Search

- Some search problems easy to specify
  - graph problems
  - Russian Farmer problem:
    - farmer, goat, cabbage cross river, while wolf following
    - How to get all three across w/out goat or cabbage being eaten

Cut

- Cut curtails backtracking
  - pred(...):- cond1,...,condk, ! ,condk+1,..., condn.
  - Always succeeds, freezes all prev. choices
- Uses of cut:
  - When get here, no other rule should be applicable
  - When get here, no hope of succeeding
  - Only 1 soln of interest, don't try generating others.
- Usually screws up reversibility

Prolog as Theorem Proving

- Facts and rules as hypotheses (universally quantified)
- Queries: ?- D(X,Y).
  - Understood as ∃X. ∃Y. D(X,Y) -- see if can prove!
- Takes hypotheses and negation of query and tries to prove contradiction.
- Constructive proof, so find X,Y s.t. D(X,Y).

Restrictions

- Horn clause logic
  - No negations in hypotheses or conclusion.
  - No disjunctions in conclusion
- Resolution is incomplete
  - Misses some proofs, so "no" may mean can't find it.
- Prolog not implement unification correctly:
  - is_own_successor(X) :- X = successor(X).
  - omits "occurs check"

Negation Problematic

- Negation based on closed-world assumption
  - ?-father(shezad,kim). *fails, but true when add fact.*
  - not(X) :- X,!,fail;
    not(X).
  - ?- not(father(shezad,kim)). *reports true*
  - *If have*
    old(X) :- not(young(X)).
    and no facts about young, then everything old.
  - *If switch then everything young.*

Evaluation of Logic Programming & Prolog

- Prolog has many faults
- Control implicit, but important
- Idea may be promising, e.g., in database
- If can ignore control and just use logic then can later optimize w/out destroying correctness.
- Very high level, self-documenting
- Efficiency is serious issue.

Today and Future

- Seems to be a bust for most part
- Still used in computational linguistics
- Constraint logic programming may have more potential -- solving lists of inequalities