

Logic Programming

Textbook, Ch. 14

1

Logic Programming

- Instead of using functions as in imperative and functional programs
- We use predicates as in predicate calculus
- Interpretation = proving theorems

2

Prolog

- Developed at Univ. of Aix-Marseille and Edinburgh in early to mid 1970s
- Goal: natural language processing and theorem proving
- Used in Japan's Fifth Generation Computing Project in 1981

3

Prolog Syntax

- Variables are uppercase
- constants, predicates are lowercase
- List syntax:
 - [1, 2, 3]
 - [head | tail]
- Program consists of
 - facts, rules, and goals

4

Facts

```
female(shelley).  
male(bill).  
female(mary).  
male(jake).  
father(bill, jake).  
father(bill, shelley).  
mother(mary, jake).  
mother(mary, shelley).
```

5

Rules

```
parent(X, Y) :- mother(X, Y).  
parent(X, Y) :- father(X, Y).  
grandparent(X, Z) :-  
    parent(X, Y), parent(Y, Z).  
sibling(X, Y) :-  
    mother(M, X), mother(M, Y),  
    father(F, X), father(F, Y).  
ancestor(X, X).  
ancestor(X, Y) :-  
    parent(X, Z), ancestor(Z, Y).
```

6

Queries

```
?- father(bill, jake).
yes
?- father(X, jake).
X = bill
yes
?- father(bill, X).
X = jake
; X = shelley
yes
```

7

Another Example

```
% Sir Bedevere's reasoning in Monty Python and
% the Holy Grail to prove that girl is a witch.
```

```
witch(X) :- burns(X), woman(X).
woman(girl).
burns(X) :- isMadeOfWood(X).
isMadeOfWood(X) :- floats(X).
floats(duck).
floats(Y) :- floats(X), !, sameWeight(X, Y).
sameWeight(duck, girl).
```

```
?- witch(girl).
```

8

List Processing Predicates

```
% member(X, L) <- X is a member of L
% append(X,Y,Z) <- Z is list consisting
% of Y appended to X
```

```
member(X, [_|_]).
member(X, [_|Ys]) :- member(X, Ys).
```

```
append([], Y, Y).
append([X|Xs], Y, [X|Zs]) :-
    append(Xs, Y, Zs).
```

9

Functional Queries

```
?- member(c, [a,b,c,d,e]).  
   yes  
?- member(f, [a,b,c,d,e]).  
   no  
?- append([a,b], [c,d,e], X).  
   X = [a,b,c,d,e]  
   yes
```

10

Relational Queries

```
?- member(e, [a,b,c,d,X]).  
   X = e  
   yes  
?- append(X, [c,d,e], [a,b,c,d,e]).  
   X = [a,b]  
   yes  
?- append([a,b], Y, [a,b,c,d,e]).  
   Y = [c,d,e]  
   yes
```

11

Relational Queries

```
?- append(X, Y, [a,b,c,d,e]).  
   X = [], Y = [a,b,c,d,e]  
; X = [a], Y = [b,c,d,e]  
; X = [a,b], Y = [c,d,e]  
   yes  
?- member(X, [a,b,c]), member(X, [c,d]).  
   X = c  
   yes
```

12

Trace of Prolog Program

```
?- member(X, [a,b,c]), member(X, [c,d]).
```

```
% first attempt  
% choice point: X = a  
member(a, [c,d])  
member(a, [d])  
member(a, [])
```

```
% fails
```

13

Trace of Prolog Program

```
% backtrack to choice point  
% second attempt  
member(X, [b,c]), member(X, [c,d])  
% choice point: X = b  
member(b, [c,d])  
member(b, [d])  
member(b, [])
```

```
% fails
```

14

Trace of Prolog Program

```
% backtrack to last choice point  
member(X, [c]), member(X, [c,d])  
% choice point: X = c  
member(X, [c,d])
```

```
% succeeds: X = c
```

15

Cut

- Prevents backtracking
- Cuts off previous choice points

```
floats(duck).  
floats(X) :-  
    floats(Y), !, sameWeight(X, Y).
```

```
member(X, [X,_]) :- !.  
member(X, [_,_]) :- member(X, _).
```

16

Execution of Prolog Programs

- Prove that goal is satisfiable
- Search of facts/rules is top-down
- Execution of sub-goals is left to right
- Closed-world assumption:
 - anything not in database is false
- Negation not equivalent to logical not
- Integer calculation, I/O don't fit well into logical proof search

17

Applications of Prolog

- Relational database queries
- Expert systems
- Parsing of context-free languages
- Natural language processing
- Teaching programming, as early as in grade school

18
