

An Introduction (I) to SML, Java & Prolog

CS216

1

Prolog

CS216

2

Logic-Oriented Style

- Relations
- Solving problems with relations!
- There are a number of dialects
 - SWI-Prolog

CS216

3

Terms

- Everything in Prolog is built from *terms*.
- Three kinds of terms:
 - Constants: integers, real numbers, atoms
 - Variables
 - Compound terms

CS216

4

Term: Constants

- Integer constants: **123**
- Real constants: **1.23**
- Atoms:
 - A **lowercase letter** followed by any number of additional letters, digits or underscores: **name**
 - But an atom is not a variable; more like a string constant.

CS216

5

Term: Variables

- Any name beginning with an **uppercase letter** or an underscore, followed by any number of additional letters, digits or underscores: **X**, **Parent**, **Child**
- Most of the variables you write will start with an uppercase letter.
 - Those starting with an underscore, including **_**, get special treatment.

CS216

6

Term: Compound Terms

- An atom followed by a parenthesized, comma-separated list of one or more terms:
 - `x(y, z), +(1, 2), .(1, []), parent(adam, seth), x(Y, x(Y, Z))`
- A compound term is not a function call: as structured data

CS216

7

Syntax of Terms

```

<term> ::= <constant>
        | <variable>
        | <compound-term>

<constant> ::= <integer>
              | <real number>
              | <atom>

<compound-term> ::= <atom> ( <termlist> )
    
```

CS216

8

Facts

```
<fact> ::= <term> .
```

- The simplest kind of thing in the database is a *fact*: a term followed by a period.
- Parent**(X,Y) = X is the parent of Y.


```
parent(kim,holly).
parent(margaret,kim).
parent(margaret,kent).
parent(esther,margaret).
parent(herbert,margaret).
parent(herbert,jean).
```

CS216

9

Rules

```

<rule> ::= <term> :- <termlist> .
<termlist> ::= <term> | <term> , <termlist>
    
```

```

      head
      |
greatgrandparent(GGP,GGC) :-
    parent(GGP,GP),
    parent(GP,P),
    parent(P,GGC).
    
```

conditions

CS216

10

Rules

```

      head
      |
greatgrandparent(GGP,GGC) :-
    parent(GGP,GP),
    parent(GP,P),
    parent(P,GGC).
    
```

conditions

- A rule says how to prove something: to prove the head, prove the conditions/

CS216

11

A Prolog Program

- A program consists of a list of *clauses*

```

<clause> ::= <fact>
            | <rule>
    
```

CS216

12

A Prolog Program

```
parent(kim,holly).
parent(margaret,kim).
parent(margaret,kent).
parent(esther,margaret).
parent(herbert,margaret).
parent(herbert,jean).

greatgrandparent(GGP,GGC) :-
    parent(GGP,GP), parent(GP,P),
    parent(P,GGC).
```

CS216

13

Example: A Prolog Program

```
% COMMENTS parent relation
parent(kim,holly).
parent(margaret,kim).
parent(margaret,kent).
parent(esther,margaret).
parent(herbert,margaret).
parent(herbert,jean).
```

- `parent(X,Y)` = X is the parent of Y.

CS216

14

SWI-Prolog Environment



- Prompting for a query with `?-`

CS216

15

The consult Predicate

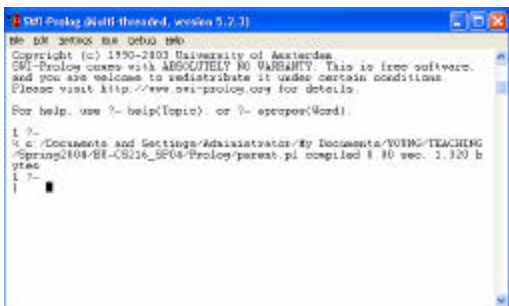
- Predefined predicate to read a program from a file into the database.
- File relations (or `relations.pl`) contains the parent facts.

```
?- consult(relations).
% relations compiled 0.00 sec, 0 bytes
Yes
?- 
```

CS216

16

The consult Predicate



CS216

17

Prolog Queries

```
?- parent(margaret,kent).
Yes
?- parent(fred,pebbles).
No
?- parent(margaret,kent)
| .
Yes
?- 
```

CS216

18

Prolog Queries

```
?- parent(P,jean).  
  
P = herbert  
  
Yes  
?- parent(P,esther).  
  
No
```

CS216

19

Queries With Variables



```
1 ?- parent(P,jean).  
2 P = herbert.  
3 Yes.  
4 ?- parent(P,esther).  
5 No.
```

CS216

20

Conjunction Queries

```
?- parent(margaret,X), parent(X,holly).  
  
X = kim  
  
Yes
```

CS216

21

Multiple Solutions

```
?- parent(margaret,Child).  
  
Child = kim ;  
  
Child = kent ;  
  
No
```

- By typing ; rather than Enter, you ask the Prolog system to find more!

CS216

22

Multiple Solutions

```
?- parent(Parent,kim),  
parent(Grandparent,Parent).  
  
Parent = margaret  
Grandparent = esther ;  
  
Parent = margaret  
Grandparent = herbert ;  
  
No
```

CS216

23

Unification

- Pattern-matching using Prolog terms
- Two terms unify if there is some way of binding their variables that makes them identical

CS216

24

Example

```
parent(kim,holly).
parent(margaret,kim).
parent(margaret,kent).
parent(esther,margaret).
parent(herbert,margaret).
parent(herbert,jean).

greatgrandparent(GGP,GGC) :-
    parent(GGP,GP), parent(GP,P), parent(P,GGC).
```

- **greatgrandparent(X,Y)** = X is the greatgrandparent of Y.

CS216

25

Query

```
?- greatgrandparent(esther,GreatGrandchild).
GreatGrandchild = holly
Yes
```

CS216

26

Rules Using Other Rules

```
grandparent(GP,GC) :-
    parent(GP,P), parent(P,GC).

greatgrandparent(GGP,GGC) :-
    grandparent(GGP,P),
    parent(P,GGC).
```

- **greatparent(X,Y)** = X is the greatparent of Y.

CS216

27

Recursive Rules

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :-
    parent(Z,Y),
    ancestor(X,Z).
```

- **X** is an ancestor of **Y** if:
 - Base case: **X** is a parent of **Y**
 - Recursive case: there is some **Z** such that **Z** is a parent of **Y**, and **X** is an ancestor of **Z**

CS216

28

The = Predicate

- The goal **=(X,Y)** succeeds if and only if X and Y can be unified:

```
?- =(parent(adam,seth),parent(adam,X)).
X = seth
Yes
?- parent(adam,seth)=parent(adam,X).
X = seth
Yes
```

- An operator is just a predicate for which a special abbreviated syntax is supported

29

Arithmetic Operators

- Predicates **+**, **-**, ***** and **/** are operators

```
?- X = +(1,*(2,3)).
X = 1+2*3
Yes
?- X = 1+2*3.
X = 1+2*3
Yes
```

CS216

30

Arithmetic Operators

```
?- +(X,Y) = 1+2*3.
```

```
X = 1  
Y = 2*3
```

```
Yes
```

```
?- 1+2 = 2+1.
```

```
No
```

```
?- 7 = 1+2*3.
```

```
No
```

CS216

31

The is Predicate

```
?- X is 1+2.
```

```
X = 3
```

```
Yes
```

```
?- 7 = 1+2*3.
```

```
No
```

CS216

32

Example: The Factorial Predicate

```
factorial(0,1).
```

```
factorial(N, F) :- N > 0, N1 is N-1,  
factorial(N1,F1), F is N*F1.
```

```
?- factorial(0,X).
```

```
X = 1
```

```
Yes
```

```
?- factorial(4,X).
```

```
X = 24
```

```
Yes
```

CS216

33

The not Predicate

```
?- member(1,[1,2,3]).
```

```
Yes
```

```
?- not(member(4,[1,2,3])).
```

```
Yes
```

- not(X) succeeds if X fails

CS216

34

Lists

```
?- X = [].
```

```
X = []
```

```
Yes
```

```
?- X = .(1,.(2,.(3,[]))).
```

```
X = [1, 2, 3]
```

```
Yes
```

```
?- X = [1,2,3].
```

```
X = [1, 2, 3]
```

```
Yes
```

CS216

35

Lists



CS216

36

List Notation With Tail

```
?- .(X, Y) = [1, 2, 3].
```

```
X = 1
```

```
Y = [2, 3]
```

```
Yes
```

```
?- [1, 2|X] = [1, 2, 3, 4, 5].
```

```
X = [3, 4, 5]
```

```
Yes
```

CS216

37

The append Predicate

- `append(X,Y,Z)` succeeds if and only if Z is the result of appending the list Y onto the end of the list X

```
append([], B, B).
append([Head|TailA], B, [Head|TailC]) :- append(TailA, B, TailC).
```

```
?- append([1, 2], [3, 4], Z).
```

```
Z = [1, 2, 3, 4]
```

```
Yes
```

CS216

38

Predicate – More than A Function

```
?-
append(X, [3, 4],
[1, 2, 3, 4]).
```

```
X = [1, 2]
```

```
Yes
```

```
?- append(X, Y, [1, 2, 3]).
```

```
X = []
```

```
Y = [1, 2, 3] ;
```

```
X = [1]
```

```
Y = [2, 3] ;
```

```
X = [1, 2]
```

```
Y = [3] ;
```

```
X = [1, 2, 3]
```

```
Y = [] ;
```

```
No
```

CS216

39

Predicate – More than A Function

CS216

40

Predicate – More than A Function

CS216

41

The reverse Predicate

- Predefined `reverse(X,Y)` unifies Y with the reverse of the list X

```
reverse([], []).
reverse([Head|Tail], X) :-
    reverse(Tail, Y),
    append(Y, [Head], X).
```

```
?- reverse([1, 2, 3, 4], Y).
```

```
Y = [4, 3, 2, 1] ;
```

```
No
```

CS216

42

The reverse Predicate

```

SWI-Prolog (Multi-threaded, version 5.2.3)
Welcome to SWI-Prolog (64-bit)
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.
For help, use ?- help(Topic). or ?- apropos(Word).

1 ?-
% c:/Documents and Settings/Administrator/My Documents/YOUNG-TEACHING/Spring2004/BU-CS216_SF04/Prolog/append.pl
compiled 0.00 sec. 888 bytes
% c:/Documents and Settings/Administrator/My Documents/YOUNG-TEACHING/Spring2004/BU-CS216_SF04/Prolog/reverse.pl
compiled 0.00 sec. 888 bytes
1 ?-
| reverse([1,2,3,4],Y).
Y = [4, 3, 2, 1]
Yes
2 ?-

```

CS216

43

The Corresponding Program in SML

Java

```

class Driver {
    public static void main(String[]
    args) {
        IntList a = new IntList(null);
        IntList b = a.cons(2);
        IntList c = b.cons(1);
        int x = a.length() +
            b.length() +
            c.length();
        a.print();
        b.print();
        c.print();
        System.out.println(x);
    }
}

```

SML

```

val a=[];
val b=2::a;
val c=1::b;
val x=(length a)+
    (length b)+
    (length c);
a;
b;
c;
x;

```

CS216

44

The Corresponding Program in SML

```

Standard ML of New Jersey 110.0.7
Standard ML of New Jersey, Version 110.0.7, September 28, 2000 [CH&MB]
- val a=[];
- val b=2::a;
- val c=1::b;
- val x=(length a)+(length b)+(length c);
- val Y=3::int;
- a;
- b;
- c;
- x;
- Y;
- GC #0.0.0.0.1.15:  (0 ms)
-

```

CS216

45

The Corresponding Program in Prolog

Java

```

class Driver {
    public static void main(String[]
    args) {
        IntList a = new IntList(null);
        IntList b = a.cons(2);
        IntList c = b.cons(1);
        int x = a.length() +
            b.length() +
            c.length();
        a.print();
        b.print();
        c.print();
        System.out.println(x);
    }
}

```

Prolog

?

CS216

46

The Corresponding Program in Prolog

```

intlist - Notepad
File Edit Format View Help
cons(X,Y,[X|Y]).

len([], 0).
len([_|_],L) :- len(_,_), L is 1 + L1.

addlen3(X,Y,Z,L) :- len(X,L1),len(Y,L2),len(
Z,L3), L is L1+L2+L3.

```

CS216

47

The Corresponding Program in Prolog

```

SWI-Prolog (Multi-threaded, version 5.2.3)
Welcome to SWI-Prolog (64-bit)
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.
For help, use ?- help(Topic). or ?- apropos(Word).

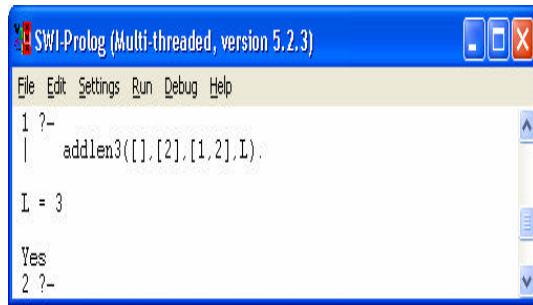
1 ?-
% c:/Documents and Settings/Administrator/My Documents/YOUNG-TEACHING/Spring2004/BU-CS216_SF04/Prolog/append.pl
compiled 0.00 sec. 888 bytes
% c:/Documents and Settings/Administrator/My Documents/YOUNG-TEACHING/Spring2004/BU-CS216_SF04/Prolog/reverse.pl
compiled 0.00 sec. 888 bytes
1 ?-
| reverse([1,2,3,4],Y).
Y = [4, 3, 2, 1]
Yes
2 ?-

```

CS216

48

The Corresponding Program in Prolog



```
SWI-Prolog (Multi-threaded, version 5.2.3)
File Edit Settings Run Debug Help
1 ?-
|   addlen3([], [2], [1, 2], I).
I = 3
Yes
2 ?-
```

CS216

49