PROLOG LISTS, OPERATORS, ARITHMETIC

Ivan Bratko

University of Ljubljana

These slides are meant to be used with a Prolog system to demonstrate the examples, and the book: I. Bratko, Prolog Programming for Artificial Intelligence, 4th edn., Pearson Education 2011. The slides alone are not self-sufficient.

PROLOG

- Prolog = “pure Prolog” + additions

- Pure Prolog ~ logic
- Additions make Prolog’s logical basis to work in practice

- Additions:
- “Pure” (do not affect logical meaning – just notational cosmetics)
- “Dirty” (do not have a logical meaning, eg. write(X))

- Some additions: list notation, operator notation, arithmetic, I/O

LIST NOTATION

- Examples of lists:

[a, b, c, d]

[ ]

[ann, tennis, tom, running]

[link(a,b), link(a,c), link(b,d)]

[a, [b,c], d, [ ], [a,a,a], f(X,Y)]

HEAD AND TAIL

- **L = [a, b, c, d]**
  - **a** is *head* of **L**
  - **[b, c, d]** is *tail* of **L**
- More notation, vertical bar:
  - **L = [Head | Tail]**
  - **L = [a, b, c] = [a | [b, c]] = [a, b | [c]] = [a, b, c | [ ]]**

LIST NOTATION IS ONLY SYNTACTIC SUGAR

- List notation: [Head | Tail]
- Equivalent to standard Prolog notation: **.(Head, Tail)**
- Note: “.” is a functor

- Equivalent terms:

**[a, b, c] = .(a, .(b, .(c, [ ]))) **

The latter expression can be, as usual, shown as a tree (first dot is root of tree)

LIST MEMBERSHIP

```
% member( X, L):  X is member of L

member( X, [ X | _ ]).              % X appears as head of list

member( X, [ _ | L])  :-
   member( X, L).                   % X in tail of list
```

TRY VARIOUS USES OF **member/2**

CONCATENATION OF LISTS

% conc(L1, L2, L3): L3 is concatenation of L1 and L2

conc([ ], L, L). % Base case

conc([X | L1], L2, [X | L3]) :- % Recursive case

conc(L1, L2, L3).

TRY MANY USES OF **conc/3**

MANY USES OF CONC

?- conc([a,b,c], [1,2,3], L).

L = [a,b,c,1,2,3]

?- conc([a,[b,c],d], [a,[ ],b], L).

L = [a, [b,c], d, a, [ ], b]

?- conc(L1, L2, [a,b,c]).

....

GENERATING LISTS OF INCREASING LENGTH

Try this:

?- conc(L, _, _).

....

Which months precede may, which follow may?

?- Months = [jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec] ,

conc(Before, [may | After], Months).

Delete everything that follows three consecutive occurrences of ‘z’

```
?- L1 = [a,b,z,z,c,z,z,z,d,e],          % Given list
   conc( L2, [z,z,z | _ ], L1).         %  L2 is L1 up to 3 z’s
```

LIST MEMBERSHIP WITH CONC

% member2(X, L): X is member of list L

```
member2( X, L) :-
   conc( _, [X | _ ], L).
```

LIST DELETION

% del(X, L, NewL)

```
del( X, [X | Tail], Tail).

del( X, [Y | Tail], [Y | Tail1] ) :-
    del( X, Tail, Tail1).

?-  del( X, [ a, b, c, d], L1).
...
```

LIST INSERTION

```
%  insert( X, L, NewL):
%     insert X into L “non-deterministically” at any position,
%     resulting in NewL

insert( X, L, [X | L]).                % Insert X as head

insert( X, [Y | L], [Y | NewL])  :-
   insert( X, L, NewL).                % Insert X into tail
```

INSERT AS OPPOSITE TO DELETE

?- del(apple, L, [1,2,3]). % What is L?

...

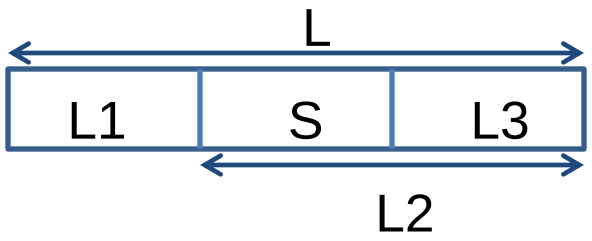
```
%  insert( X, L, LongerL):  Insert X in L at any position, giving LongerL
insert( X, List, LongerList)  :-
   del( X, LongerList, List).

%  member3( X, L):  X is element of L, alternative implementation
member3( X, L)  :-
   del( X, L, _).              % X can be deleted from L
```

SUBLIST OF A LIST

% sublist(List, Sublist): Sublist appears as a sublist in List

% It's easy!

% Just draw List and Sublist and rewrite the drawing into Prolog



sublist(S, L) :-

conc(L1, L2, L),

conc(S, L3, L2).

OPERATOR NOTATION

OPERATOR NOTATION

- Operator notation is just a cosmetic, surface notational improvement
- Equivalent notations for arithmetic expressions:

  +(*(2,a), *(b,c)) = 2*a + b*c

- +, * are infix operators built into Prolog
- Convention in Prolog: + has higher precedence than *

USER CAN INTRODUCE HER OWN OPERATORS

has(peter, information).

supports(floor, table).

This can be rewritten with operators as:

:- op(600, xfx, has).

:- op(600, xfx, supports).

peter has information.

floor supports table.

TYPES OF OPERATORS

(1) infix operators

xfx **xfy** **yfx**

(2) prefix operatos

fx **fy**

(3) postfix operators

xf **yf**

- **yfx** is left associative operator
- **xfy** is right associative operator
- What is the difference between **fx** in **fy**?

DECLARATION OF AN OPERATOR, “DIRECTIVE”

- **op(Prioriteta, Tip, Operator).**

- Declare appropriate operators so the the following clauses will become a legal notation in Prolog:

  mary has talent and big hopes and many high ambitions

  ?- X = ..., Y = ..., ...

  if X > Y then Z = X else Z = Y.

- Write an interpreter in Prolog for if-then-else statement.

ARITHMETIC

BUILT-IN PREDICATES FOR ARITHMETIC OPERATIONS

- Try to add 1 + 2 with:

?- X = 1 + 2.

X = 1 + 2 % Prolog just keeps expression unevaluated

- This is better:

?- X is 1 + **2.** % “is”: built-in predicate that forces calculation

X = **3**

ARITHMETIC OPERATIONS

- +, -, *, /, ** addition, subtraction, ...
- //, mod operations on integers
- sin, cos, log, ... standard functions

?- X is 2 + sin(3.14/2).

X = 2.9999996829318345

?- A is 11/3.

Y = 3.6666666666666665

?- B is 11//3.

C = 3

?- C is 11 mod 3.

C = 2

COMPARISON PREDICATES

X	>	**Y**	
X	<	**Y**	
X	>=	**Y**	
X	=<	**Y**	
X	=:=	**Y**	X and Y are numerically equal
X	=\=	**Y**	X and Y are not numerically equal

?- 315 * 3 >= 250*4.

yes

?- 2+5 = 5+2.

no

?- 2+5 =:= 5+2.

yes

LENGTH OF LIST

% length(L, N): N is the length of list L

```
length( [ ], 0).

length( [ _ | L], N)  :-
  length( L, N0),
  N  is  N0 + 1.
```

In the second clause, can the order of goals be reversed?