Lists

York University CSE 3401
Vida Movahedi
Overview

• Definition and representation of Lists in Prolog
  – Dot functor

• Examples of recursive definition of predicates
  – islist,
  – member, delete
  – append, multiple,
  – prefix, suffix, sublist

[ref.: Clocksin- Chap.3 and Nilsson- Chap. 7]
[also Prof. Gunnar Gotshalks’ slides]
Lists

• A list:
  – is an ordered sequence of elements that can have any length.
  – It is a term.
  – Either an empty list [] or it has a head X and a tail L represented as [X|L] where X is a list item and L is a list.
  – List notation in Prolog: [a, b, c, d, ...]

• The dot:
  – is a functor for representing lists with two arguments, the head and the tail of a list
  – A list of one element [a] is [a| [] ] implemented in Prolog as .(a, [])
  – [a, b] is .(a, .(b, []))

• Note [a, b, c] is not the same as [a, [b,c]]
[a, b, c] is .(a, .(b, .(c, [])))
Examples

• Write the Prolog definition for being a list.

  islist([]).
  islist([Head|Tail]) :- islist(Tail).

• Write the Prolog definition for being a member of a list.

  member(X, [X|L]).
  member(X, [Y|L]) :- member(X,L).
Examples (cont.)

:- member(3, [2, 3, 4, 5]).
true

:- member(3, [2, [3, 4], 5]).
false

Our definition does not consider members of members (nested lists)

:- member(X, [1, 2]).
X = 1 ;
X = 2 ;
false

Unlike other programming languages, inputs can be unknowns

:- member(2, L).
L = [2 |_] ;
L=[_, 2 | _];
...

Note the recursive definition of member
Recursive Search

- Example:
  member(X, [X|L]). : boundary condition
  member(X, [Y|L]) :- member(X,L). : recursive case
  member(X, [Y|L]) :- member(X,L).

  :-member(X, [a,b,c]).

  X = a; 
  X = b; 
  X = c; 
  false
- delete(X, L1, L2) is true if L2 is the result of deleting X from L1 (just once).

  - For example: delete(5, [1, 5, 4, 2], [1, 4, 2]).

\[
delete(X, [X|L], L).
\]
\[
delete(X, [Y|L], [Y|L1]) :- delete(X, L, L1).
\]
Append

• Join two lists:
  Example: append([1,2], [3,4], [1,2,3,4])

  append([], L, L). : boundary condition
  append([X|L1], L2, [X|L3]) :- append(L1, L2, L3). : recursive case
  a smaller problem

• Possible Queries:
  [Nilsson]
  :- append([a, b], [c, d], [a, b, c, d]).

  true

  :- append([a, b], [c, d], X).

  X=[a, b, c, d]

  or even
  :- append(Y, Z, [a, b, c, d]).
append([], X, X).
append([X|Y], Z, [X|W]) :-
  append(Y, Z, W).

:- append(Y, Z, [a, b, c, d]).

\[
\begin{align*}
Y &= [] & Z &= [a, b, c, d] \\
Y &= [a] & Z &= [b, c, d] \\
Y &= [a, b] & Z &= [c, d] \\
Y &= [a, b, c] & Z &= [d] \\
Y &= [a, b, c, d] & Z &= []
\end{align*}
\]
Example: multiple occurrences in a list

- multiple(L) is true if L is a list with multiple occurrences of some element [Nilsson]:

  \[
  \text{multiple}([\text{Head} \mid \text{Tail}]) :- \text{member}(\text{Head}, \text{Tail}).
  \text{multiple}([\text{Head} \mid \text{Tail}]) :- \text{multiple}(\text{Tail}).
  \]

- Writing multiple(..) using append(..)

  \[
  \text{multiple}(L) :- \text{append}(L1, [X \mid L2], L), \text{append}(L3, [X \mid L4], L).
  \]

What is missing in definition of multiple(..)? How can it be corrected?
Prefix/ Suffix with append

- Write prefix(P,L) which is true if P is a prefix of L.
  
  \[ \text{prefix(P, L):} \text{ append(P, \_, L).} \]
  
  - Is [] a prefix of L?

- Write suffix(S,L) which is true if S is a suffix of L
  
  \[ \text{suffix(S, L):} \text{ append(\_, S, L).} \]

- Exercise: Try writing prefix and suffix without using append.
More Examples with append

• sublist(S,L) is true if S is a sublist of L
  – in other words, S is the suffix of a prefix
  – Using append(..):

\[
\text{sublist}(S,L) :- \text{append}(_, S, \text{Left}), \text{append}(\text{Left}, _, L).
\]
More Examples with append

• Re-writing delete(X,L1,L2) with append(..):

\[\text{delete}(X, L, R):- \text{append}(L1,[X|L2],L), \text{append}(L1, L2, R).\]
Append is expensive!

\[
\text{append}([], L, L). \\
\text{append}([X|L1], L2, [X|L3]) :- \text{append}(L1, L2, L3).
\]

- The complexity of appending two lists, \( L_1 \) and \( L_2 \), is \( O(n) \) where \( n \) is the length of the first list.

- Consider \( \text{reverse}(L, R) \) defined as:
  \[
  \text{reverse}([], []). \\
  \text{reverse}([X|L], R) :- \text{reverse}(L, L1), \text{append} (L1, [X], R).
  \]

- Complexity of \( \text{reverse}(..) \) is \( O(n^2) \) where \( n \) is the length of \( L \).