ITSE301 Logic Programming

List Processing

14-5-2024

Lists in Prolog

- A list is a collection of items, not necessarily homogeneous.
- In Prolog, lists are inbuilt data structures.
- Lists can be used to represent sets, stacks, queues, linked lists, and several complex data structures such as trees, graphs, etc.

Lists in Prolog

- A list in Prolog is an ordered collection of items denoted as $[i_1, i_2, \ldots, i_n]$.
- prolog lists allow direct access of the first element only which is denoted as Head.
- We can write a prolog list as :

[Head | Rest], where Rest is the rest of the list excluding the first element Head.

Lists in Prolog

- [] % empty list
- [a] % singleton list
- [hello, world] % 2 element list
- [[1,2,3,4], p, this] % 3 element list
- [[[1, 2], [3, 4]], [[a, b], [x, y]]]. % nested list (3 level nesting)

List Membership

- **X is a member of L if**
  - **X is the head of L, or**
  - **X is in the tail of L.**

Operations on Prolog Lists
List Membership

- **X is a member of L if**
  - **X is the head of L, or**
  - **X is in the tail of L.**
- **Thus we write:**

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

The length of a list

- **IF L is a list then it size can be recursively calculated as:**
  - The length of the empty list is 0.
  - The length of the list whose head is H and whose tail is the list T is: 1 + (the length of T).

The length of a list

- **IF L is a list then it size can be recursively calculated as:**
  - The length of the empty list is 0.
  - The length of the list whose head is H and whose tail is the list T is: 1 + (the length of T).

- **Thus we write:**

```
length([],0).
length([H|T],N) :-
        length(T,N1), N is N1+1.
```

The sum of a list

- **We observe that**
  - **The sum of the empty list is 0.**
  - **The sum of the list whose head is H and whose tail is the list T is: H + (the sum of T).**

The sum of a list

- **We observe that**
  - **The sum of the empty list is 0.**
  - **The sum of the list whose head is H and whose tail is the list T is: H + (the sum of T).**

- **Thus we write:**

```
sum([],0).
sum([H|T],Sum) :-
        sum(T,Sum1), Sum is Sum1 + H.
```

The sum of a list

- **We observe that**
  - **The postivesum of the empty list is 0.**
  - **The postivesum of the list whose head is postive H and whose tail is the list T is: H > 0, H + (the postivesum of T).**
  - **The sum of the list whose head is –H and whose Tail is T is: the sum of the Tail.**

- **Thus we write:**

```
sum([],0).
sum([H|T],Sum) :-
        sum(T,Sum1), Sum is Sum1 + H.
```

The size of a list

- **We observe that**
  - **The size of the empty list is 0.**
  - **The size of the list whose head is H and whose tail is the list T is: 1 + (the size of T).**

- **Thus we write:**

```
size([],0).
size([H|T],N) :- size(T,N1), N is N1+1. % or

size([_|T],N) :- size(T,N1), N is N1+1.
```

The reverse of a list

- **we write:**

```
reverse(List, Reversed) :-
                        reverse(List, [], Reversed).
reverse([], Reversed, Reversed).
reverse([Head|Tail], SoFar, Reversed) :-
    reverse(Tail, [Head|SoFar], Reversed).
```

Exercises

Let L be any list of terms. Define Prolog predicates for the following:

- average(L,N) is true if N is the average of all the numbers in L, or just 0 if the sum is 0
- sumpos(L,N) is true if N is the sum of all the *positive* numbers in L
- sumsquare(L,N) is true if N is the sum of the squares of all the numbers in L
- maxlist(L,N) is true if N is the largest element in the list L.
- maxpos(L,N) is true if N is the position of the largest element in the list L. (If there's more than one occurrence of the maximum, then this should be the *first* position at which it appears.)
- last(L,E) is true if E is the last element of L
- evenpos(L) which prints out the elements of L at positions 2,4,6... up to the end of the list (Use write/1 to print out the elements.)