CSEN403: Concepts of Programming Languages
Logic Programming IV

Prof. Dr. Slim Abdennadher
Dr. Nada Sharaf

Spring Semester 2021

Lists in Prolog

- A useful data structure
- A built-in structure in Prolog.
- Prolog is un-typed so you can have a list of **any** terms
- e.g.

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


?= X=[Y,3, s(0)].
x=[Y,3, s(0)]
```

Internal Representation

- `[H|T]`
- `?- [1,2,6]=[H|T].`
  `H=1, T=[2,6]`
- `[8,4,9,2]=[H1,H2|Tail].`
  `H1=8, H2=4, Tail=[9,2]`
- `?- [H|T] = [1, [2,3]].`
  `H=1, T=[[2,3]]`

Lists: Member

Define a predicate `mem/2`. `mem(E,L)` is true if `E` is on of the elements inside `L`. Examples:

```
?- mem(6,[7,2,6]).
true
?- mem(10,[6,2,9]).
false
```

`E` is a member in a list if

- It is the first element
- It occurs in the rest of the list

```
mem(El,[H|T]):- El=H.

mem(El,[H|T]):- mem(El, T).
```

Append

Define a predicate app/3. app(L1,L2,L3) is true if L3 is the result of appending L2 to L1. Examples:

```
?- app([1,4,2],[9,10],L).
L=[1,4,2,9,10]

app([],L,L).
app([H|T],L,R):-
                app(T,L,R1),
                R=[H|R1].
```

Reverse

Define a predicate `rev/2`. `rev(L1,L2)` is true if `L2` contains the same elements of `L1` in reversed order. Examples:

```
?- rev([1,4,2],L).
L=[2,4,1]

rev([],[]).
rev([H|T],List):-
                rev(T,T1),
                app(T1,[H],List).
```

Mytsery

What is the functionality of `mys/3`.

```
mys(X,[X|L],L).
mys(X,[_|L],R) :- mys(X,L,R).
```

More on Lists

Implement a predicate `insert/3`. `insert(X,L1,L2)` is true if `L2` is contains all the elements in `L1` in addition to `X` i.e. `L2` is the result of inserting `X` into `L1`.

```
insert(X,[],[X]).
insert(X,[H|T],[X,H|T]).
insert(X,[H|T],[H|NT]):- insert(X,T,NT)
```

Accumulator Technique

- Has an imperative flavour
- Accumulate intermediate results in an extra attribute.
- At every point, accumulator contains a partial result.
- After all elements are processed, the accumulator contains the final result

Accumulator Technique

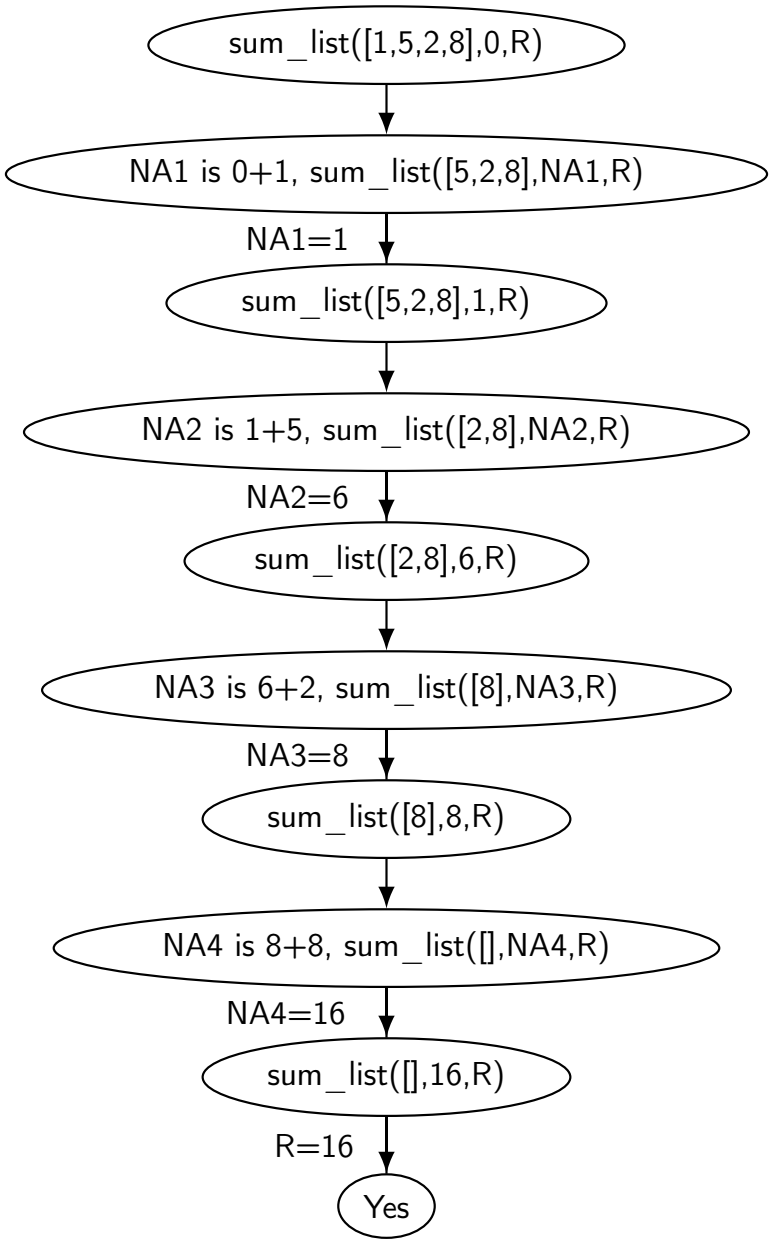
- Has an imperative flavour
- Accumulate intermediate results in an extra attribute.
- At every point, accumulator contains a partial result.
- After all elements are processed, the accumulator contains the final result

First Example: Sum the elements of a List

List	Result So Far	Final Result
[1,5,2,8]	0	?
[5,2,8]	0+1=1	?
[2,8]	1+5=6	?
[8]	6+2=8	?
[]	8+8=16	16

```
sum_list([], ResultSoFar, FinalResult):-
                          FinalResult=ResultSoFar.
sum_list([H|T], ResultSoFar, FinalResult):-
                          NewAcc is ResultSoFar + H.
                          sum_list(T,NewAcc,FinalResult).
```

Trace


sum_list([1,5,2,8],0,R)
NA1 is 0+1, sum_list([5,2,8],NA1,R)
NA1=1
sum_list([5,2,8],1,R)
NA2 is 1+5, sum_list([2,8],NA2,R)
NA2=6
sum_list([2,8],6,R)
NA3 is 6+2, sum_list([8],NA3,R)
NA3=8
sum_list([8],8,R)
NA4 is 8+8, sum_list([],NA4,R)
NA4=16
sum_list([],16,R)
R=16
Yes

sum_list/2

```
sum_list(L,S):- sum_list(L,0,S).
```

reverse

List	Intermediate Result	Final Result
[1,2,3]	[]	?
[2,3]	[1]	?
[3]	[2,1]	?
[]	[3,2,1]	?

```
reverse([],I,I).
reverse([H|T],I,R):- reverse(T,[H|I],R).
reverse(H,R):- reverse(H,[],R).
```

Set of Answers

- Collect objects together
- `setof(Things, Condition, Bag)`
- `?- setof(X, member(X,[4,3,7,1]),List).`
  `List=[4,3,7,1]`

Gathering Results

```
male(king_george_v).
male(king_edward_viii).
male(king_george_vi).
male(prince_philip).
male(prince_charles).
male(prince_andrew).
male(prince_edward).
male(prince_william).
male(prince_harry).
```

What is the output of

```
?- setof(X,male(X),L).
L = [king_edward_viii,king_george_v,king_george_vi,
prince_andrew,prince_charles,prince_edward,
prince_harry,prince_philip,prince_william].
```

Another Example

Who are the sons of queen_mary?

```
setof(X,(male(X),parent(queen_mary,X)),L).
```

Thank you