Universidade Federal de Santa Maria
Centro de Tecnologia
ELC117 - Paradigmas de Programação
Profª Andrea Schwertner Charão

Listas em Prolog

1 Introdução

- Estruturas de dados suportadas nativamente em Prolog
- Sequência de qualquer número de itens
- Podem conter itens de mesmo tipo ou de tipos diferentes

2 Representação

- Elementos entre colchetes e separados por vírgula

- Representação básica com enumeração dos elementos - exemplos:

```
[azul, verde, amarelo]
['Fulano', 35]
[0.6, 0.7, azul]
[casa(1, azul), casa(2, verde)]
```

- Usando listas em fatos e consultas - exemplos:

  Fato (declarado em arquivo):

```
idades([[andre,25], [jose,30]]).
```

  Consulta:

```
?- idades([[_,A], [_,B]]).
A = 25,
B = 30.
```

- Representação mais genérica:

  [H | T], onde

  - H representa o primeiro elemento da lista (head)
  - T representa o restante da lista, sem o primeiro elemento (tail)

  Exemplos:

```
primeiro([P | _], P).
ultimo([U], U).
ultimo([_ | R], U) :- ultimo(R, U).

?- primeiro([a,b,c,d], P).
P = a.

?- primeiro([9, 8, 7], X).
X = 9.

?- ultimo([9,8,7],X).
X = 7 .

?- ultimo([], P).
false.
```

3 Exemplo no depurador (SWI-Prolog)

```
?- trace.
true.

[trace]  ?- ultimo([a,b,c,d],U).
   Call: (6) ultimo([a, b, c, d], _G379) ? creep
   Call: (7) ultimo([b, c, d], _G379) ? creep
   Call: (8) ultimo([c, d], _G379) ? creep
   Call: (9) ultimo([d], _G379) ? creep
   Exit: (9) ultimo([d], d) ? creep
   Exit: (8) ultimo([c, d], d) ? creep
   Exit: (7) ultimo([b, c, d], d) ? creep
   Exit: (6) ultimo([a, b, c, d], d) ? creep
U = d .
```

4 Predicados úteis com listas

Notação dos argumentos

- argumento de entrada: +
- argumento de saída: -
- argumento de entrada/saída: ?

Predicados

- **length**(?List, ?Int): verdadeiro se Int representa o número de elementos da lista List
- **member**(?Elem, ?List): verdadeiro se Elem é membro da lista List
- **sort**(+List, -Sorted): verdadeiro se Sorted é equivalente à lista List ordenada; elementos duplicados são removidos do resultado
- **append**(?List1, ?List2, ?List1AndList2): verdadeiro se List1AndList2 é a concatenação de List1 e List2
- **nextto**(?X, ?Y, ?List): verdadeiro se Y segue X em List
- **nth0**(?N, ?List, ?Elem): verdadeiro se Elem é o N-ésimo elemento de List (índices começando em 0)
- **nth1**(?N, ?List, ?Elem): verdadeiro se Elem é o N-ésimo elemento de List (índices começando em 1)
- **last**(?List, ?Last): verdadeiro se Last é o último elemento de List

Exemplos

Predicado length

```
?- length([a,b], X).
X = 2.

?- length([a,b], 3).
false.
```

Predicado member

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

?- member(4, [1,2,3]).
false.

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

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

Predicado sort

```
?- sort([a,z,b], X).
X = [a, b, z].

?- sort([4,1,9], X).
X = [1, 4, 9].

?- sort(['Fulano','Beltrano'], X).
X = ['Beltrano', 'Fulano'].

?- sort([4,1,9,1], X).
X = [1, 4, 9].

?- sort([2,1],[1,2]).
true.

?- sort([2,1,1],[1,2]).
true.

?- sort([2,1,1,2],[1,2]).
true.

?- sort([2,1,1,2],[2,1,1,2]).
false.
```

Predicado append

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

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

```
?- append(A, B, [1,2,3,4]).
A = [],
B = [1, 2, 3, 4] ;
A = [1],
B = [2, 3, 4] ;
A = [1, 2],
B = [3, 4] ;
A = [1, 2, 3],
B = [4] ;
A = [1, 2, 3, 4],
B = [] ;
false.
```

Predicado nextto

```
?- nextto(3, 4, [1,2,3,4]).
true .

?- nextto(4, 3, [1,2,3,4]).
false.

?- nextto(3, 4, [1,2,3,0,4]).
false.
```