Representação do Conhecimento e Raciocínio
Programação Lógica
Prolog – parte 3

Inteligência Artificial - 2020/1

Prolog – Programas com Listas

- Concatenação

  Conc(L1, L2, L3)

  Estruturação do problema:

- Se L1 é lista vazia, o resultado da concatenação é igual a L2

- Se L1 não é vazia, é da forma [X|L]. O resultado da concatenação é [X|LR] onde LR é a concatenação de L com L2.

Prolog – Programas com Listas

- Concatenação
  - Programa:

```
conc([ ], L, L).

conc([X|L1], L2, [X|L3]) :-
                    conc(L1, L2, L3).
```

Prolog – Programas com Listas

- Concatenação
  - Programa:

```
conc([ ], L, L).

conc([X|L1], L2, [X|L3]) :-
                    conc(L1, L2, L3).
```

  - Consultas:

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

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

?- conc([a,b,c,20], [3, z| Z], L).
L = [a, b, c, 20, 3, z|Z].
```

Prolog – Programas com Listas

- Concatenação
  - É possível consultar esse programa na forma inversa: decompor uma dada lista em duas sublistas

```
?- conc(L1, L2, [a, b, c]).
L1=[]
L2=[a,b,c] ;
L1=[a]
L2=[b,c] ;
L1=[a,b]
L2=[c] ;
L1=[a,b,c]
L2=[] ;
false.
```

```
?- conc([a,b],L1,[a,b,c,d]).
L1 = [c, d].
```

```
?- conc([X,Y],L1,[a,b,c,d]).
X = a,
Y = b,
L1 = [c, d].
```

Prolog – Programas com Listas

- Adicionar um elemento como último elemento de uma lista:
  - Programa:

```
add_ultimo(X,[ ],[X]).

add_ultimo(X, [X1|Y],[X1|L]) :-
                        add_ultimo(X,Y,L).
```

Prolog – Programas com Listas

- Adicionar um elemento como último elemento de uma lista:
  - Programa:

```
add_ultimo(X,[ ],[X]).

add_ultimo(X, [X1|Y],[X1|L]) :-
                        add_ultimo(X,Y,L).
```

  - Consultas:

```
?- add_ultimo(x,[a,b,c],L).
L=[a,b,c,x];
false.

?- add_ultimo([g,h],[f,d,i],L).
L = [f,d,i,[g,h]] ;
false.

?- add_ultimo(X,Y,[a,b,c]).
X = c ,
Y = [a,b] ;
false.
```

Prolog – Programas com Listas

- Eliminar um elemento de uma lista:
  - Programa:

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

del(X,[Y|Cauda],[Y|Cauda1]) :-
                    del(X,Cauda, Cauda1).
```

Prolog – Programas com Listas

- Eliminar um elemento de uma lista:
  - Programa:

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

del(X,[Y|Cauda],[Y|Cauda1]) :-
                    del(X,Cauda, Cauda1).
```

  - Consultas:

```
?- del(a,[a,b,a,c],L).
L=[b,a,c] ;
L=[a,b,c] ;
false.

?- del(a,[x,sf,fe,[d,a,c]],L).
false.
```

Prolog – Programas com Listas

- Somar os elementos de uma lista numérica
  - Programa:

```
soma([ ],0).
soma([Elem| Cauda], S) :- soma (Cauda,S1),
                          S is S1 + Elem.
```

  - Consulta:

```
?- soma([1,2,3,4,5,6], S).
S = 21.
```

Prolog – Programas com Listas

- Contar os elementos de uma lista
  - Programa:

```
conta([ ],0).
conta([ _ | Cauda], N) :- conta(Cauda, N1),
                          N is N1 + 1.
```

  - Consulta:

```
?- conta([1,2,3,4,5,6],C).
C = 6.
```

Prolog – Programas com Listas

- Eliminar todas as ocorrências de um elemento de uma lista

  - Programa:

```
del_todas(Elem,[ ],[ ]).
del_todas(Elem, [Elem|Y], Z) :-   del_todas(Elem,Y,Z).
del_todas(Elem,[Elem1|Y], [Elem1|Z]) :- Elem \== Elem1,
                                        del_todas(Elem,Y,Z).
```

  - Consulta:

```
?- del_todas(a, [a,b,a,c],L).
L=[b,c];
false.
```

Prolog – Programas com Listas

- Retirar todas a repetições de uma lista
  - Programa:

```
retirar_rep([ ],[ ]).
retirar_rep([Elem|Cauda],[Elem|Cauda1]) :-
        del_todas(Elem,Cauda,Lista),
        retirar_rep(Lista,Cauda1).
```

  - Consulta:

```
?- retirar_rep([a,b,[a],c,b,x,p1,b,a,[a]],Resultado).
Resultado=[a,b,[a],c,x,p1];
false.
```

Prolog – Programas com Listas

- Contar o número de ocorrências de um dado elemento no primeiro nível de uma lista:

  - Programa:

```
conta_occor(Elem,[ ],0).
conta_occor(Elem,[Elem|Y],N) :-
                              conta_ocorr(Elem,Y,N1),
                              N is N1 + 1.
conta_occor(Elem,[Elem1|Y], N) :-
                               Elem \== Elem1,
                               conta_ocorr(Elem,Y,N).
```

Prolog – Programas com Listas

- Contar o número de ocorrências de um dado elemento no primeiro nível de uma lista:
  - Consulta:

?- conta_occor(5,[a, [b,5,c], 5, z, b, 5, par(a,b)],C).

C = 2.

```
conta_occor(Elem,[ ],0).
conta_occor(Elem,[Elem|Y],N) :-
                        conta_ocorr(Elem,Y,N1),
                        N is N1 + 1.
conta_occor(Elem,[Elem1|Y], N) :-
                        Elem \== Elem1,
                        conta_ocorr(Elem,Y,N).
```

- Próxima aula:
- Uso do predicado “corte”
- Entrada e saída