Predicado de Concatenação

- Concatenar duas listas, formando uma terceira:
  1. Se o primeiro argumento é a lista vazia, então o segundo e terceiro argumentos devem ser o mesmo
  2. Se o primeiro argumento é a lista não-vazia, então ela tem uma cabeça e uma cauda da forma [X|L1]; concatenar [X|L1] com uma segunda lista L2 resulta na lista [X|L3], se L3 é a concatenação de L1 e L2

Predicado de Concatenação

- Concatenar duas listas, formando uma terceira:
  - Se o primeiro argumento é a lista vazia, então o segundo e terceiro argumentos devem ser o mesmo

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

  - Se o primeiro argumento é a lista não-vazia, então ela tem uma cabeça e uma cauda da forma [X|L1]; concatenar [X|L1] com uma segunda lista L2 resulta na lista [X|L3], se L3 é a concatenação de L1 e L2

Predicado de Concatenação

- Concatenar duas listas, formando uma terceira:
  - Se o primeiro argumento é a lista vazia, então o segundo e terceiro argumentos devem ser o mesmo

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

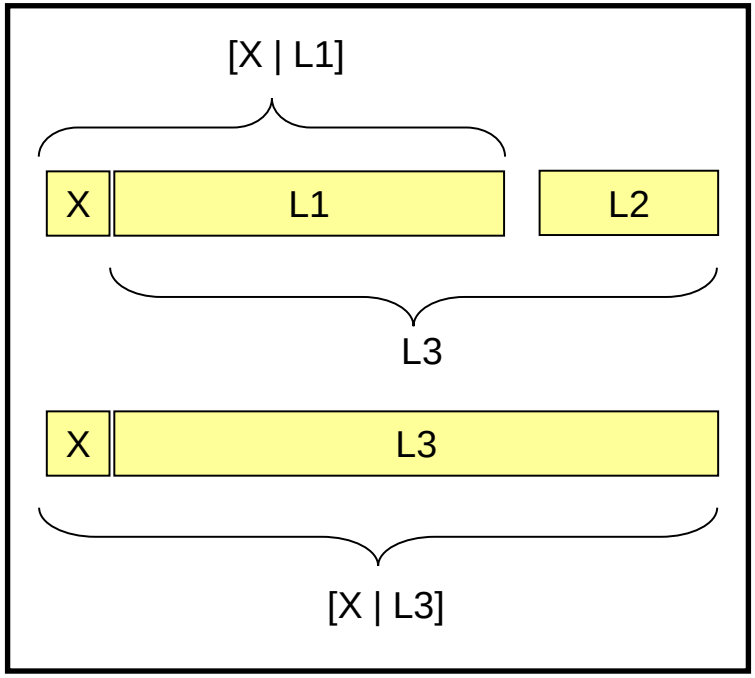
  - Se o primeiro argumento é a lista não-vazia, então ela tem uma cabeça e uma cauda da forma [X|L1]; concatenar [X|L1] com uma segunda lista L2 resulta na lista [X|L3], se L3 é a concatenação de L1 e L2

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

Predicado de Concatenação

- Programa completo:

```
% concatenar(?/+L1,?/?L2,+/?L)
concatenar([ ],L,L).
concatenar([X|L1],L2,[X|L3]) :-
   concatenar(L1,L2,L3).
```



Exercícios

- **Definir** uma nova versão do predicado **último**, que encontra o último elemento de uma lista, utilizando a **concatenação** de listas

```
ultimo(X,[X]).
ultimo(X,L):-
            concatenar(_,[X],L).

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

Exercícios

- **Definir** predicado **penúltimo**

```
penultimo(X,[X,_]).
penultimo(X,[_|C]):-
            penultimo(X,C).
```

Exercícios

- Encontrar o **comprimento** de uma lista (Sugestão: A is A1+1)

```
comprimento([],0).
comprimento([X|C],A):-
                        comprimento(C,A1),
                        A is A1+1.
```

Exercícios

- **Retirar** elementos de uma lista

```
retirar(E,[E|C],C).
retirar(E,[E1|C],[E1|C1]) :-
                              retirar(E,C,C1).
```

Slides baseados em:

Bratko, I.;
Prolog Programming for Artificial Intelligence,
3rd Edition, Pearson Education, 2001.

Clocksin, W.F.; Mellish, C.S.;
Programming in Prolog,
5th Edition, Springer-Verlag, 2003.

Programas Prolog para o
Processamento de Listas e Aplicações,
Monard, M.C & Nicoletti, M.C., ICMC-USP, 1993

Material elaborado por José Augusto Baranauskas
Adaptado por Huei Diana Lee