HAC

LISTAS

- LISTA é uma seqüência ordenada de elementos.
  - Pode ter qualquer comprimento.
  - Elementos de listas podem ser simples ou estruturados (inclusive listas)
- **Exemplos:**

  [ ] (lista vazia)
  [a, b, c]
  [maria, joao, pedro, carlos]
  [1, 329, -15, par(a,b), X, [2, c, Y], 2000]

HAC

Listas

- Listas são divididas em:
  - cabeça - primeiro elemento
  - cauda - o que resta tirando o primeiro elemento
- Exemplos:

[a, b, c]
cabeça: a
cauda: [b,c]

[X, Y, 234, abc]
cabeça: X
cauda: [Y, 234, abc]

HAC

- As partes da lista são combinadas pelo funtor • (ponto):

• (Cabeça, Cauda)

[a, b, c] equivale a

• (a, • (b, • (c, [ ])))

HAC

Padrão de Listas:

A barra vertical separa a cabeça da cauda.

[X|Y] representa listas com pelo menos um elemento

[X,Y | Z] representa listas com pelo menos dois elementos

Símbolos antes da barra são ELEMENTOS
Símbolo após a barra é LISTA

Unificação de listas

- Os padrões de listas são muito utilizados nas operações de unificação
  - Lista 1: [a1, a2, a3, a4]
  - Lista 2: [X | Y]
  - Resultados:
    - X = a1
    - Y = [a2, a3, a4]

Lista 1	Lista 2	Resultado
[a1, a2, a3, a4]	[X | Y]	X = a1 Y = [a2, a3, a4]
[a1]	[X | Y]	X = a1 Y = [ ]
[ ]	[X | Y]	não unifica
[[a, b] c, d]	[X | Y]	X = [a, b] Y = [c, d]
[[ana, Y] | Z]	[[X, foi], ao, cinema]	X = ana Y = foi Z = [ao, cinema]
[[ana, Y] | Z]	[[X, foi], [ao, cinema]]	X = ana Y = foi Z = [[ao, cinema]]
[a, b, c, d]	[X, Y | Z]	X = a Y = b Z = [c, d]
[ana, maria]	[X, Y | Z]	X = ana Y = maria Z = [ ]
[ana, maria]	[X, Y, Z]	não unifica

Operações sobre listas

- Operações sobre listas freqüentemente usam busca recursiva.
- São o mecanismo principal para programação em Prolog
- O programa é construído com base nas duas partes da lista: *cabeça e cauda*.

Exemplos

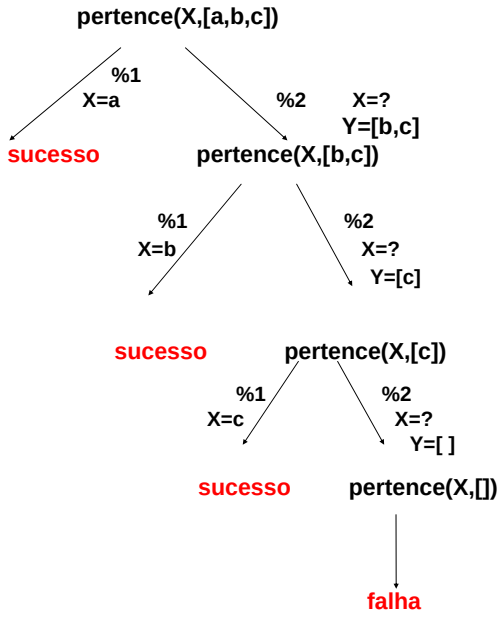
- **Verificar se um elemento é membro de uma lista.**
- **Estruturação da solução:**
  - **X é membro de L se:**
  - **X é a cabeça de L, ou**
  - **X é membro da cauda de L**
- **É necessário definir:**
  - **nome : pertence**
  - **parâmetros: elemento, lista**

Programa “pertence”

```
pertence(X, [X| _ ]).                % cláusula 1

pertence(X, [ _ | Y]) :- pertence(X,Y).
                                      % cláusula 2

/*  _  representa variável anônima  */
```


Após definir o programa, é possível consultá-lo:

```
?- pertence(a, [1,2,a,c,b]).
yes

?- pertence(a, [1,2,3]).
no

?- pertence (X, [a,b,c]).
X = a ;
X = b ;
X = c ;
no
```

Concatenação

conc(L1, L2, L3)

- **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.**

conc([ ], L, L).

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

- **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=[] ;
no

?- conc(_, [Mes1, maio, Mes2| _],
[jan, fev, mar,abr,maio,jun,jul, ago, set, out, nov, dez]).

Mes1=abr
Mes2=jun ;
No

?- conc(L1, [E1,E2,E3], [a,b,c,[a,b],d,[ ]]).

L1 = [a,b,c] ,
E1 = [a,b] ,
E2 = d ,
E3 = [ ] ;
no

Adicionar um elemento como último elemento de uma lista:

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

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



```
?- add_ultimo(x,[a,b,c],L).
L=[a,b,c,x];
no

?- add_ultimo([g,h],[f,d,i],L).
L = [f,d,i,[g,h]] ;
no

?- add_ultimo(X,Y,[a,b,c]).
X = c ,
Y = [a,b] ;
no
```


Eliminar um elemento de uma lista:

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

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

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

?- del(a,[x,sf,fe,[d,a,c]],L).
no
```


Convenção de notação

- Definição de Predicados
  pred(+Arg1, ?Arg2, -Arg3)
- Modo de Declaração

- \+ Argumento de entrada. Deve estar instanciado quando o predicado é chamado
- \- Argumento de saída. Deve ser uma variável não instanciada quando o predicado é chamado. Se o predicado der sucesso, será instanciada ao valor retornado
- ? Argumento de entrada ou de saída. Pode estar instanciado ou não.