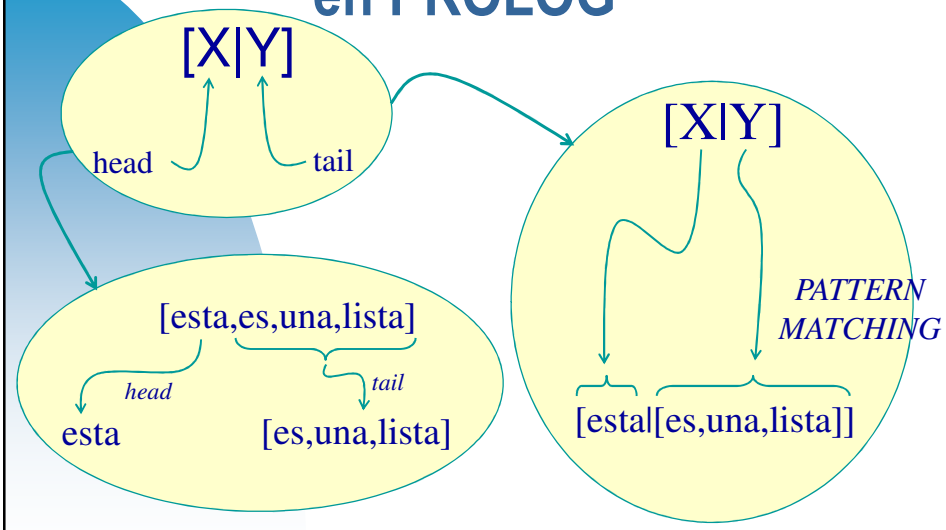
Lenguajes de programación

Listas en PROLOG

RECORDATORIO Predicados en PROLOG

- La sintaxis de un predicado es
  `pred(arg1, arg2, …, argn)`
- Donde los argumentos pueden ser
  - Constantes
    - Numéricas
    - Atómicas
  - Variables
    - Con nombre
    - Sin nombre
  - **Estructuras**
    - Funciones
    - **Listas**
    - …

Representación de listas en PROLOG



Ejemplos

- Sea la lista:

```
L = [[1,2,3],[4,5,6],7,8,9]
```

```
[X,Y|Z]
```

```
X = [1,2,3]
Y = [4,5,6]
Z = [7,8,9]
```

```
[[X|Y],Z|W]
```

```
X = 1
Y = [2,3]
Z = [4,5,6]
W = [7,8,9]
```

Ejemplos

- Sea la lista:

```
L = [[1,2,3],[4,5,6],7,8,9]
```

Pensamiento "Lógico"

- Consiste en razonar estableciendo cuando un predicado es verdadero (o falso).
- Funciones Predefinidas en Prolog:
  - **member(X,Y)**
    - Verdadero cuando X es elemento de Y.
  - **length(X,Y)**
    - Verdadero cuando Y es el número de elementos de la lista X.
  - **append(X,Y,Z)**
    - Verdadero cuando Z es la concatenación de X e Y.

Pensamiento Recursivo

- Aplican las mismas reglas que se emplean en la programación funcional.
- **Caso base (fin de recursión).**
  - Sobre el número cero.
  - Sobre la lista vacía.
  - …
- **Caso general ("llamado" recursivo).**
  - Sobre el número anterior a n (n-1).
  - Sobre la lista ([X|Y]) sin su primer elemento (Y).

Ejemplo: Combinando los dos "pensamientos"

- Programar el predicado `miembro(X,Y)` que es verdadero cuando X aparece en la lista Y.
- **Consultas:**

```
?- miembro(3,[1,2,3,4]). → Yes
?- miembro(5,[1,2,3,4]). → No
?- miembro(X,[1,2,3,4]).
        → X = 1 ;
        → X = 2 ;
        → X = 3 ;
        → X = 4
```

Programa

- **Caso base**: X es elemento de una lista cuyo primer elemento es X.

```
miembro(X,[X|_]) :- !.
```

- **Caso general**: X es elemento de una lista si X aparece en el resto (el "tail") de la lista.

```
miembro(X,[_|Z]) :-
  miembro(X,Z).
```

Ejemplo

- Programar el predicado `longitud(X,Y)` que es verdadero cuando Y es la cantidad de elementos de X.
- **Consultas:**

```
?- longitud([1,2,3,4],4).
        → Yes
?- longitud([1,2,3,4],2).
        → No
?- longitud([1,2,3,4],N).
        → N = 4
```

Programa

- **Caso general:** La longitud de una lista de forma [X|Y] es N si N es la longitud de Y más uno.

```
longitud([_|Y],N) :-
    !, longitud(Y,N1),
    N is N1+1.
```

- **Caso base**: La longitud de una lista vacía es cero.

```
longitud([],0).
```

Ejemplo

- Programar el predicado concatena(X,Y,Z) que es verdadero cuando Z es la concatenación de las listas X y Y.
- **Consultas:**

```
?- concatena([1,2,3],[a,b],[1,2,3,a,b]).
        → Yes
?- concatena([1,2,3],[a,b],Z).
        → Z = [1,2,3,a,b]
?- concatena(X,Y,[1,2,3]).
        → X = [], Y = [1,2,3] ;
        → X = [1], Y = [2,3] ;
        → X = [1,2], Y = [3] ;
        → X = [1,2,3], Y = []
```

Programa

- **Caso general**: La concatenación de una lista de forma [X|Y] con otra lista L es otra lista [X|L1] si L1 es la concatenación de Y con L.

```
concatena([X|Y],L1,[X|L2]) :-
      !, concatena(Y,L1,L2).
```

- **Caso base**: La concatenación de una lista vacía con otra lista es igual a esta última.

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

Otro Ejemplo

- Intersección de 2 conjuntos (listas):

```
interseccion([X|Y], Z, W):-
   not(miembro(X, Z)), !,
   interseccion(Y, Z, W).

interseccion([X|Y], Z, [X|W]):-
   !, interseccion(Y, Z, W).

interseccion([], _, []).
```