Estructuras de Datos en PROLOG

UNIVERSIDAD DE VALLADOLID

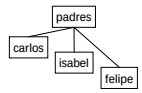
Departamento de Informática

Índice:

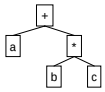
Estructuras y árboles

- Una estructura de datos en PROLOG se representa mediante un árbol

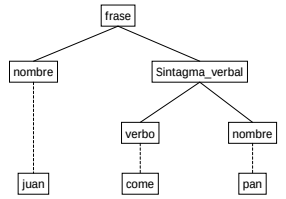
padres(hijo, madre, padre)
padres(carlos, isabel, felipe)



a+b*c



frase(nombre, sintagma_verbal(verbo, nombre))



frase(nombre(juan), sintagma_verbal(verbo(come), nombre(pan)))

Listas (I)

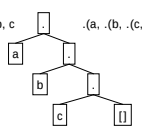
- Lista es una secuencia ordenada de términos (constantes, variables, estructuras, e, incluso, otras lista).
- La lista es utilizada: análisis sintáctico, gramáticas, diagramas de flujo, grafos, etc.
- Manejo específico de listas -> LISP.
- La lista es un caso particular de estructura en PROLOG => recursividad en la definición.

Listas (II)

- Tipo particular de árbol: cabeza y cola
  - Funtor es “.”
  - El final de la lista es “[]”

Lista de un solo elemento:
.(a, [])

Lista: a, b, c .(a, .(b, .(c,[])))



Representación habitual de las listas:
[a,b,c]
[los, hombres, [van, a, pescar]]

Cabeza es el primer término.
Cola es una lista que contiene al resto.

Un forma de instanciar a una lista con cabeza X y cola Y sería:
[X|Y]

Pertenencia (I)

- Saber si un objeto pertenece a lista.
  [carlos_i, felipe_ii, felipe_iii, felipe_iv, carlos_ii]
- Construir el predicado “miembro”:
  - miembro(X, [X|_]). ⇔ miembro(X, [Y|_]):- X=Y.
    (sólo comprueba coincidencia con la cabeza)
  - añadir: miembro(X, [_|Y]) :- miembro(X,Y).
- Verifica la coincidencia con la cola y de forma recursiva va operando sobre otra lista progresivamente más pequeña:
  ?- miembro(felipe_ii, [carlos_i, felipe_ii, felipe_iii, felipe_iv, carlos_ii]).
  ?- miembro(X, [carlos_i, felipe_ii, felipe_iii, felipe_iv, carlos_ii]).

Pertenencia (II)

- Evitar definiciones circulares:

```
padre(X, Y) :- hijo(Y, X).
hijo(A, B) :- padre(B, A).
```

(se entra en un bucle infinito)

- Recursión por la izquierda:

```
persona(X) :- persona(Y), madre(X, Y).
persona(adan).

?- persona(X).
```

Intercambiar el orden de las líneas del programa

Práctica: alterar frase

- Diálogo:
  - Usuario: tu eres un ordenador
  - Prolog: yo no soy un ordenador
  - Usuario: hablas francés
  - Prolog: no_hablo alemán ("_" simula a ",")
- Reglas:
  - Aceptar frase en formato lista: [tu, eres, un ordenador]
  - Cambiar cada *eres* por un *no soy*.
  - Cambiar cada *hablas* por un *no_hablo*.
  - Cambiar cada *francés* por un *alemán*.

Solución: alterar frase

```
cambiar(tu, yo).
cambiar(eres, [no, soy]).
cambiar(hablas, [no_,
    hablo]).
cambiar(frances, aleman).
cambiar(X,X).

alterar([], []).
alterar([H|T], [X|Y]):-
        cambiar(H,X),
        alterar(T,Y).
```

```
?- alterar([tu, hablas, frances], X).

X = [yo, [no_, hablo], aleman] ;

X = [yo, [no_, hablo], frances] ;
X = [yo, hablas, aleman] ;
X = [yo, hablas, frances] ;
X = [tu, [no_, hablo], aleman] ;
X = [tu, [no_, hablo], frances] ;
X = [tu, hablas, aleman] ;
X = [tu, hablas, frances] ;
```

Práctica: ordenación alfabética (I)

- Las palabras se considerarán una lista de números (enteros) correspondiente a su secuencia ASCII.
- Obtener la lista asociada a cada palabra:

```
?- name(pala, X).
X=[112, 97, 108, 97].
```

- Comparación:

```
amenor(X, Y) :- name(X, L), name(Y, M), amenorx(L, M).
amenorx([], [_|_]).
amenorx([X|_], [Y|_]) :- X<Y.
amenorx([A|X], [B|Y]) :- A=B, amenorx(X, Y).
```

Práctica: ordenación alfabética (II)

- Ejercicios:
  - ¿qué sucede si se elimina la relación amenor([], [_|_])?
  - Modifíquese el fichero PROLOG para admitir dentro de "amenor" el caso de dos palabras iguales.

Predicado "append"

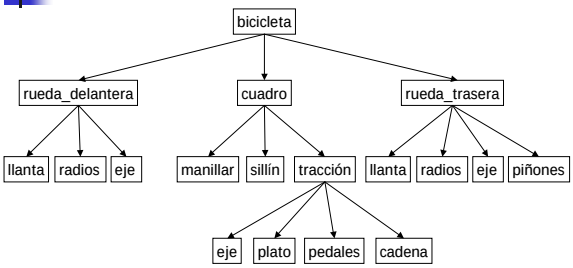
- Predicado predefinido:

```
?- append([a, b, c], [1, 2, 3], X).
X=[a, b, c, 1, 2, 3]
?- append(X, [b, c, d], [a, b, c, d]).
X=[a]
?- append([a], [1, 2, 3], [a, 1, 2, 3]).
Yes
?- append([a], [1, 2, 3], [alfa, beta, gamma]).
No
```

- Definición:

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

Práctica: inventario de piezas (I)



Práctica: inventario de piezas (II)

- Definir el árbol mediante las relaciones:
  - pieza_basica(cadena).
  - ensamblaje(bicicleta, [rueda_delantera, cuadro, rueda_trasera]).
- Construir relaciones “piezas_de”, que sirva para obtener la lista de piezas básicas para construir una determinada parte de (o toda) la bicicleta.