Universidad Tecnológica Nacional
FACULTAD REGIONAL CORDOBA

PARADIGMAS DE PROGRAMACION

Unidad V
Paradigma Lógico
Parte IV

CONTENIDOS ABORDADOS

- Objetos compuestos.
- Recursividad.
- Listas.
- Ejercitación práctica.

Objetos compuestos o estructuras

- Es una forma de introducir datos estructurados en Prolog.
- Los objetos compuestos están formados por un functor y un conjunto de argumentos.
- Los hechos con objetos compuestos son de la forma:

  predicado (argumento, functor (argumento1, argumento2, …, argumentoN)).
- Ejemplos: evento(nombre, fecha(día,mes,año), lugar)

  evento('JEIN',fecha(11,10,18), 'UTN-FRC').

  evento('CAEDI',fecha(29,09,18), 'Holiday Hotel').

Objetos compuestos

Los objetos compuestos pueden usarse como:

- **Descriptores:**
  - persona(nombre(Juan), apellido(Perez), dni(123456789))
- **Clasificadores:**

  a) lugar →trabaja(persona,lugar)

  panaderia(nombre, numero).

  taller(nombre, area, seccion).

  trabaja(juan, taller(limpieza, norte, 1)).

  b) producto → compra(persona,producto)

  producto(nombre).

  producto(nombre, marca)

  compra(juan, producto(tomate, arcor)).

Objetos compuestos

Caso de estudio

Tabla 1: Vendedores.

Legajo	Nombre	Apellido
1	Francesca	Luchetti
2	Vera	Petro
3	Lara	Tana

Tabla 2: Ventas realizadas

Código venta	Código vendedor	Importe facturado	Fecha de venta			Tipo de cobro		
						Tarjeta de crédito		Efectivo
			Día	Mes	Año	Número de cupón	Cantidad de cuotas	Débito automático (Si/No)
111	1	355.5	23	09	2013	610	3	-
222	2	230.45	25	09	2013	-	-	Si
333	1	1834.25	05	10	2013	410	4	-

Objetos compuestos

Ejemplo

%HECHOS
%venta/5
%venta(CódigoVenta,LegajoVendedor,importeFacturado, fechaVenta,TipoCobro)
%fechaVenta/3 - fechaVenta(Día,Mes,Año)
%TipoCobro: tarjeta/2, efectivo/1.
%tarjeta/2 - tarjeta(NroCupón,CantidadCuotas)
%efectivo/1 - efectivo(DebitoSiNO)
venta(111,1,355.5,fechaVenta(23,9,2013),tarjeta(610,3)).
venta(222,2,230.45,fechaVenta(25,9,2013),efectivo('si')).
venta(333,1,1834.25,fechaVenta(23,8,2013),tarjeta(410,4)).
venta(666,3,500,fechaVenta(25,10,2013),efectivo('-')).

Recursividad

- La recursión es una técnica que se basa en definir relaciones en términos de ellas mismas.
- Si del lado derecho de una cláusula aparece en algún punto el mismo predicado que figura del lado izquierdo, se dice que la cláusula tiene llamado recursivo, es decir **“se llama a sí misma”** para verificar que se cumple esa misma propiedad como parte de la condición que define a la regla, y sobre algún posible valor de sus variables.
- En la definición recursiva, es necesario considerar dos casos:
  - **Caso Básico:** Momento en que se detiene el proceso o computo, se produce el corte.
  - **Caso Recursivo:** Suponiendo que ya se ha solucionado un caso más simple, se descompone el caso actual hasta llegar al caso más simple.

Recursividad

Ejemplo: Obtener el factorial de un número.

Formulación:

0! = 1

n! = n*n-1*n-2*n-3*…..*1

n! = n*(n-1)!

(n-1)!= (n-1) * (n-2)!

….

Término general: factorial(numero, resultado)

clausulas:

%hecho, caso básico

factorial(0,1).

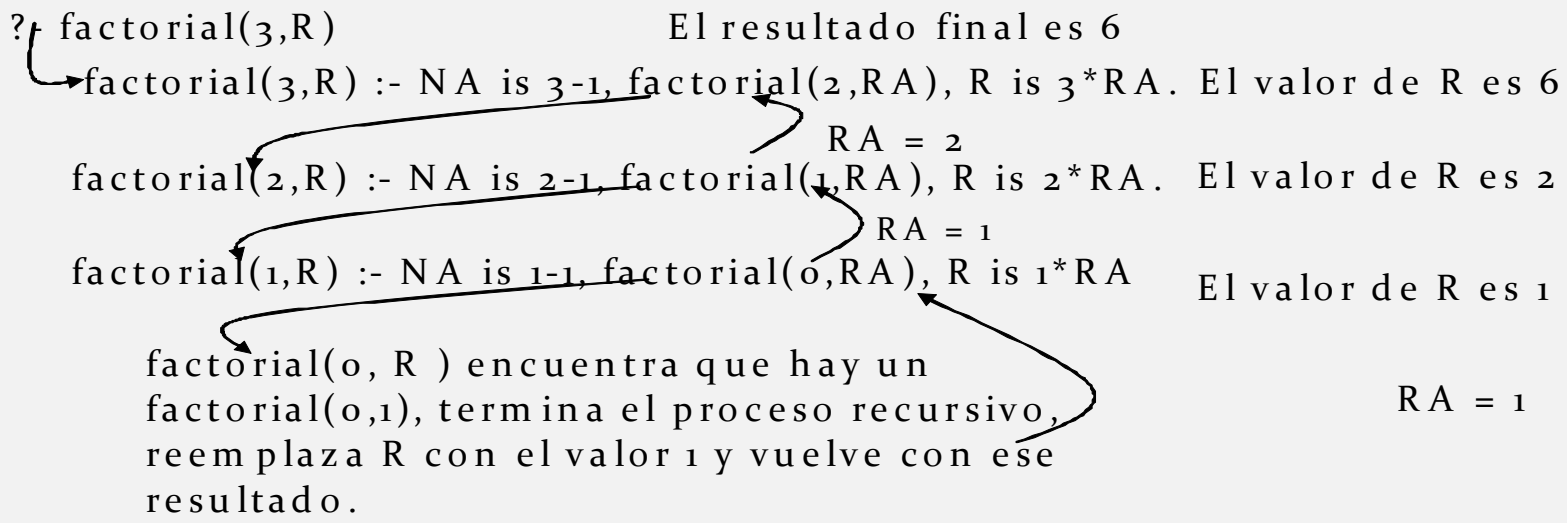
%regla, caso recursivo (N=numero, NA=numero anterior)

factorial(N,R) :- NA is N-1, factorial(NA,RA), R is N*RA.

Recursividad

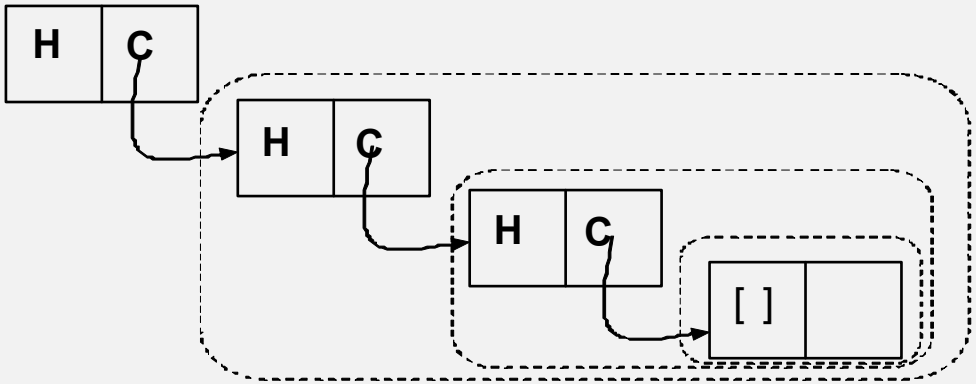
Ejemplo: Obtener el factorial de un número.

Consulta:



Listas

- Son estructuras de datos que almacenan y manipulan conjunto de términos.
- Se trata de un par ordenado donde cada componente es un término, una lista o el término NIL (lista vacía []).
- El primer componente de la lista se llama *cabeza* de la lista y la segunda *cola.*



Listas

- Por definición son recursivas, el segundo término es una lista, que a su vez posee cabeza y cola, y así se repite la construcción hasta que se encuentra la lista vacía [].
- **Definición simple**: Conjunto de términos encerrados entre corchetes y separados por coma (,).

  Por ejemplo : [1,2,3,4,5].
- **Definición formal**:

  <Lista> := [<H> | <T>]. Donde:

  <H>:= <término> ó <lista>

  <T>:= <término> ó <lista>

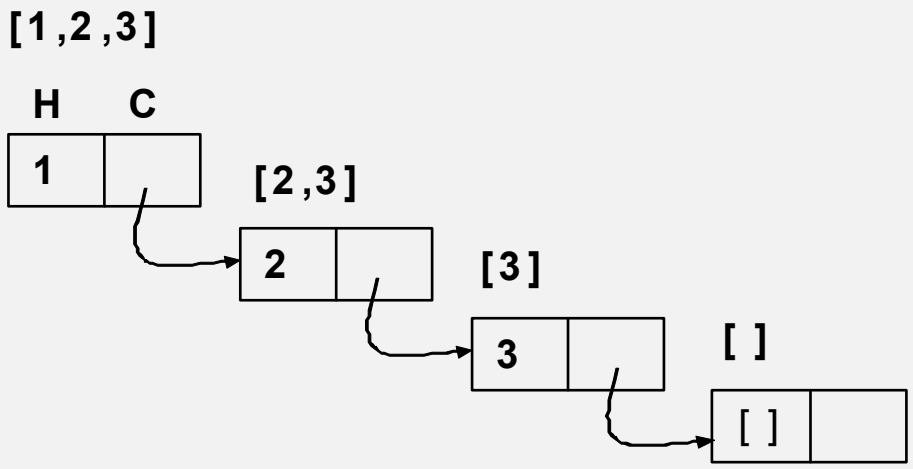
  Por ejemplo : [1, [a, b, [c]], 3, [pepe], []] es la lista cuya primera componente o cabeza es el termino constante 1 y la cola es la lista [[a, b, [c]], 3, [pepe], []]

Listas

Ejemplos:

- La siguiente lista de términos : [1, 2, 3] se representa como:

  [1 | [2 | [3 | []]]] donde el termino constante 1 es la cabeza y la cola es la lista [[2 | [3 | []]]].



Listas

Ejemplos del proceso de unificación en listas.

[X,Y] y [a,b] unifican con {X=a, Y=b}

[X,Y] y [[1,2,3],[4,5]] unifican con {X=[1,2,3], Y=[4,5]}

[X|Y] y [1,2,3] unifican con {X=1, Y=[2,3]}.

[X,Y] y [letra(a)|[letra(b)|letra(c)]]. Unifican con {X=letra(a), Y=[letra(b),letra(c)]}

- [X|Y] y [] no unifican dado que la primera lista es una lista con al menos un elemento y la lista vacía no tiene elementos.
- [X,Y | Z] unifica con cualquier lista que tenga al menos dos elementos, es decir unifica, por ejemplo con:
  - [1, 2], las variables quedan ligadas: {X=1, Y=2, Z=[] }
  - [1, 2, 3] las variables quedan ligadas: {X=1, Y=2, Z=[3] }

Listas

Otro ejemplo:

Se tiene una base de hechos con las materias que cursa cada alumno:

Hechos

cursa(juan, [matematicas, historia, computacion]).

cursa(ana, [matematicas, arte, historia]).

cursa(pedro, [geologia, logica, geografia]).

cursa(maria, [logica, arte, computacion]).

cursa(jose, [logica, historia, geografia]).

Consultas

? cursa(juan, L)

L = [matematicas, historia, computacion]

Listas

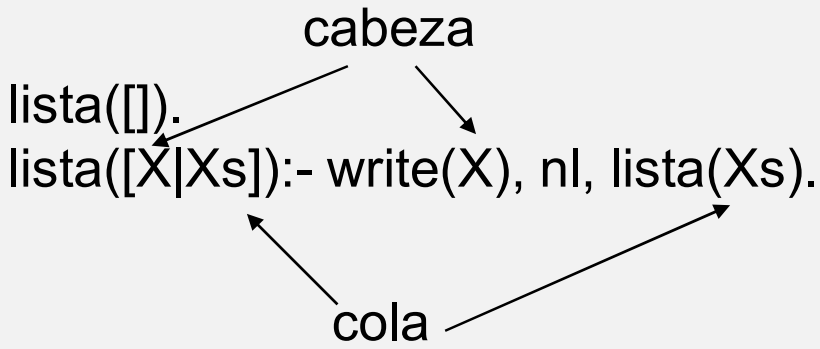
Recursividad en listas

- La definición de la lista es recursiva, podemos decir que una lista es:

  la lista vacía o

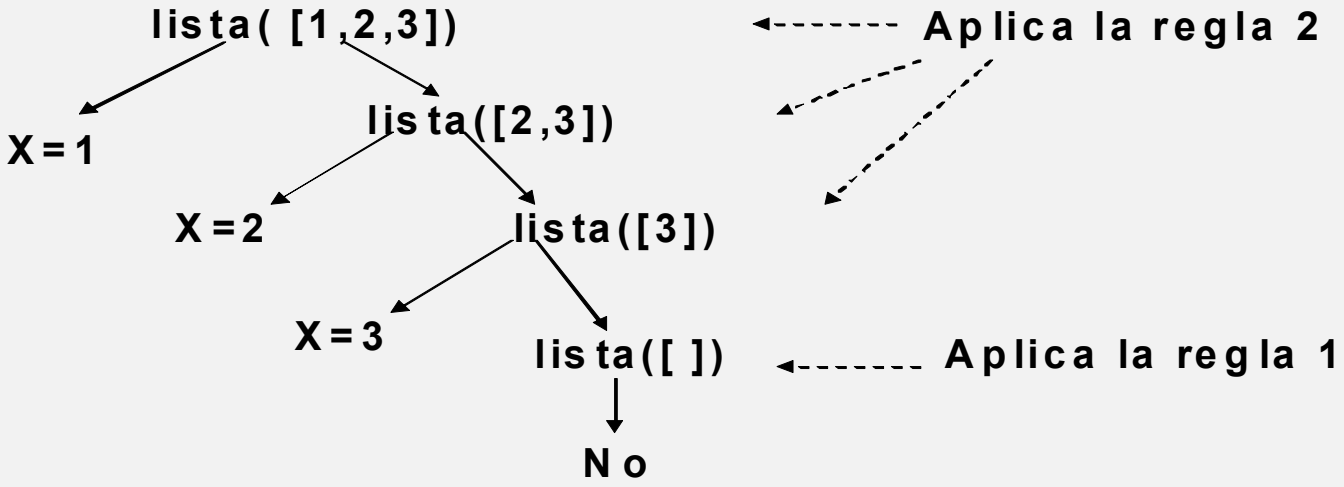
  un par donde el segundo componente es una lista.

- Un ejemplo para mostrar los elementos de una lista:



Listas

- Si tuviéramos la consulta ?- lista([1,2,3]), el árbol de prueba sería:



Listas

Miembro de una lista

- Una regla elemental es determinar si un elemento pertenece o no a la lista. Para ello, debemos programar un predicado miembro que dado un elemento nos responda “yes” si pertenece a la lista y “no”en otro caso. El predicado sería:

  miembro(X, [X | _]).

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

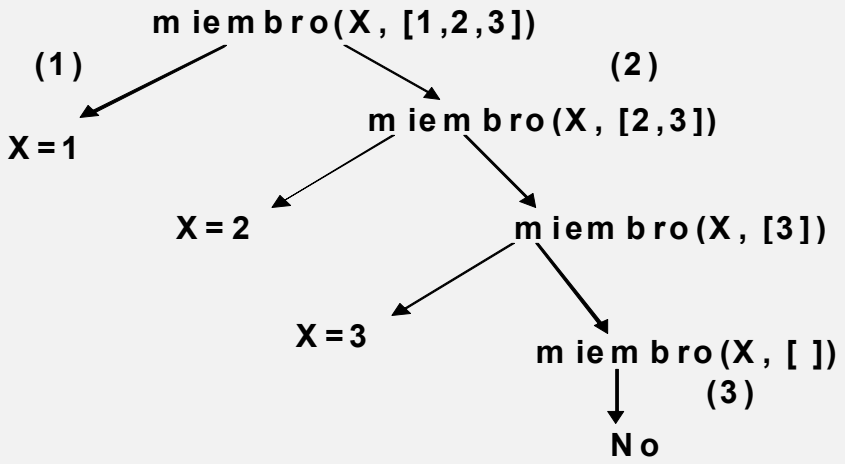
- La primera regla : Indica X es miembro de la lista que tiene X como cabeza.
- La segunda expresa: X es miembro de una lista si X es miembro del resto de dicha lista.

Listas

Miembro de una lista

?- miembro(1, [1,2,3]).
Yes

?- miembro(X,[1,2,3]).
X=1;
X=2;
X=3;
no



Listas

Sumar los elementos de una lista en forma recursiva:

- Para sumar los términos de una lista debemos asegurarnos que su contenido sea numérico.
- Se utilizan dos argumentos uno que lleva la lista a sumar y otro el resultado a obtener.
- El caso básico se da cuando se encuentra la lista vacía [], y el resultado que no afecta la suma es 0.
- El predicado sería:

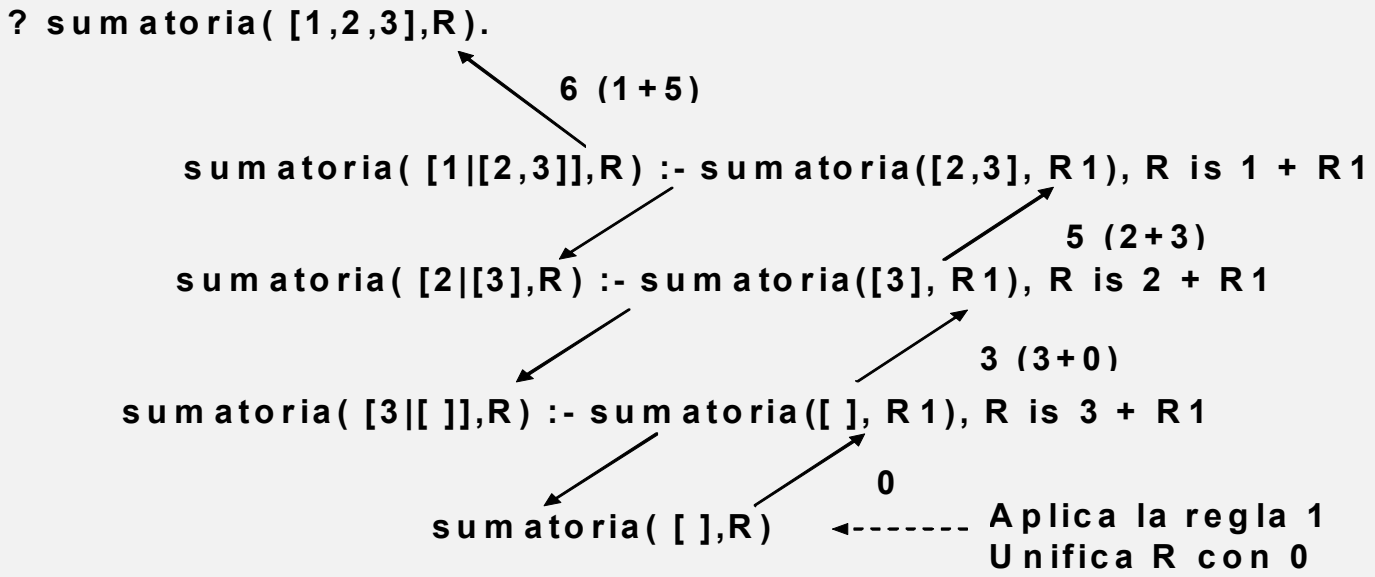
```
%sumatoria(Lista, Resultado)
%hecho: caso básico
sumatoria([],0).
%regla: caso recursivo
sumatoria([H|T], R) :- sumatoria(T, R1), R is H + R1.
```

Listas

- Ejemplo: Sumar los elementos de una lista en forma recursiva:

```
sumatoria([],0).
sumatoria([H|T], R) :- sumatoria(T, R1), R is H + R1.
```

- Consulta ?- sumatoria([1,2,3],R) el árbol de prueba sería:



Listas

Predicados de SWI-Prolog para manipular listas :

- **length(UnaLista, Longitud):** Longitud es el número de elementos de la lista UnaLista.
- **member(UnElemento, UnaLista):** Retorna true si UnElemento es un elemento de UnaLista.
- **sort(UnaLista, ListaOrdenada):** ListaOrdenada es la lista ordenada de los elementos de UnaLista sin duplicados.
- **sumlist(UnaLista, UnaSuma):** Retorna en UnaSuma la sumatoria de todos los elementos de UnaLista.
- **append(UnaLista, OtraLista, ListaConcatenada):** ListaConcatenada es la concatenación de UnaLista y OtraLista.

Listas

Predicados de SWI-Prolog para manipular listas :

- **findall(UnaVariable,UnObjetivo,UnaLista):** Colecciona todas las soluciones de un objetivo y retorna una lista como resultado con los valores del objetivo definidos por una variable.
    1. **UnaVariable**: especifica qué argumento del predicado va ser coleccionado en la lista.
    2. **UnObjetivo:** es el predicado objetivo del cual los valores serán coleccionados en la Lista.
    3. **UnaLista:** lista resultado con los valores obtenidos por **UnaVariable** en cada objetivo evaluado.

```
alumnos(L):-findall(N,alumno(N),L). -→[pedro, ana, juan]
alumnos(L,C):-findall(N,alumno(N),L), length(L,C). -→3
alumnos(LO):-findall(N,alumno(N),L), sort(L,LO).
→[ana,juan,pedro]
ejemplo2(L):-findall([N,A], alumno(_, N,A), L).
```