Prolog
Estructuras, Recursividad, Listas

Estructuras

- Mecanismo para representar tipos de datos compuestos.

Ejemplo:

```
% BD
cumpleaños(luz, fecha(20,abr,1985)).
cumpleaños(pedro, fecha(12,ene,1980)).
cumpleaños(lina, fecha(15,feb,1982)).
```

```
?- cumpleaños(pedro,X).
X = fecha(12, ene, 1980).

?- cumpleaños(X,fecha(_,ene,_)).
X = pedro ;
false.

?- cumpleaños(X,fecha(_,feb,_)).
X = lina ;
false.
```

Recursividad

- Prolog permite la definición de procedimientos recursivos. Estos siempre poseen algunas reglas no recursivas (casos base), más algunas reglas recursivas.
- Una regla recursiva es aquella que contiene cláusulas que hacen referencia a si misma.

Ejemplo de predicado recursivo

- Tomemos un árbol genealógico.
- El predicado X es descendiente de Y se define así:

```
descendiente(X,Y) :- hijo(X,Y), !.
descendiente(X,Y) :- hijo(X,Z), descendiente(Z,Y).
```

- Supongamos además los siguientes hechos:

```
hijo(b,a).
hijo(c,a).
hijo(d,b).
hijo(e,b).
hijo(f,e).
hijo(q,p).
hijo(r,q).
```

Ejercicio:
Analizar como se evaluan
`descendiente(f,a).`
`descendiente(e,c).`

Ejemplo: Cálculo del factorial

- El factorial se define recursivamente así
  - factorial(0) es 1
  - factorial(N) es n*factorial(n-1)
- Esta definición recursiva se traduce inmediatamente en un procedimiento recursivo:

```
% factorial/2
% Verdadero si arg2 es el factorial de arg1

factorial(0,1) :- !.
factorial(N,X) :- NN is N-1, factorial(NN,Y), X is Y*N.
```

Ciclos infinitos

- La recursividad puede dar origen a programas que no terminan.

```
conexionDirecta(a,b).
conexionDirecta(b,a).
conexionDirecta(b,c).
conexionDirecta(c,b).
conexion(X,Y) :- conexionDirecta(X,Y).
conexion(X,Y) :- conexionDirecta(X,Z), conexion(Z,Y).
```

- Por ejemplo las siguientes consultas no terminan:

```
?- conexion(a,b).
?- conexion(a,c).
?- conexion(a,d).
```

Ciclos

- Ya habíamos visto que cualquier ciclo se puede representar en forma recursiva:

```
while g(x) A
W(x) = (if g(x) A; W(x))
```

Ejemplo:

- Escribir la lista de enteros entre M, N

```
listaEnteros(M,N) :- M>N, nl.
listaEnteros(M,N) :- M=<N, write(M), write(','),
                     NuevoM is M+1, listaEnteros(NuevoM, N).
```

Ejemplo: Cálculo de la serie aritmética

- Obsérvese que la serie aritmética de la forma:

  A + (A+D) + … + (A+D*N)

  se puede definir recursivamente así:

  - Para N=0 la serie suma A
  - En los demás casos, el valor de la suma es la serie hasta el término N-1, más el término N.

```
% serieAritmetica/4
% Calcula la serie aritmetica con parametros A,D hasta N
% Verdadero si la suma de la serie es S

serieAritmetica(A,_,0,A).
serieAritmetica(A,D,N,S) :-   N>0, NN is N-1,
                              serieAritmetica(A,D,NN,SS),
                              S is SS+A+D*N.
```

Listas

- Una lista es una secuencia ordenada de términos. Los términos pueden ser constantes, variables, estructuras u otras listas.
- Algunos ejemplos de listas:
  - [magdalena, juan, lucas]
  - [5, 7, cuadrado, fecha(3,ene,2010)]
  - [a, [b,c], 5, [casa,perro], [3,2,1]]
  - []
- [] es la lista vacía

Construcción recursiva de listas

- Prolog utiliza las definiciones recursivas de listas:
  - [] es una lista
  - Si a es un término y B es una lista, entonces [a|B] es una lista.
  - Nada más es una lista.
- En esta definición, el primer elemento a se llama la ***cabeza de la lista***.
- La lista B con los restantes elementos se llama la ***cola de la lista***.

Ejemplos de Listas

- [x,y,z] tiene por cabeza x, y por cola [y,z]
- [x | [y,z]] es equivalente a [x,y,z]
- [x,y,z]=[A|B] se unifica con el unificador A=x, B=[y,z]
- [] = [A|B] no unifican. La lista no tiene cabeza.
- [x] = [A|B] unifica con A=x y B=[].
- [x|A] = [B|[y,z]] tiene por unificador A=[y,z], B=x.

Ejercicios

- Hacer un procedimiento que determine el primer elemento de una lista:

```
primerElemento(X, [X|_]).
```

- Hacer un procedimiento que determina la cola de una lista:

```
colaDeLista(X, [_|X]).
```

- Hacer un procedimiento que determine si dos listas comienzan por el mismo término:

```
empiezanIgual([X|_], [X|_]).
```

Diseño de Procedimientos Recursivos

- Definir el caso base, y establecer las reglas apropiadas para manejar el caso base.
- Definir las reglas para los casos recursivos, normalmente tratando un subproblema del problema original.
- Por ejemplo, para problemas con listas, el problema original se puede descomponer en dos: Lo que se hará con la cabeza de la lista y lo que se hará con la cola.

Ejercicios

- Hacer un procedimiento que devuelta el último elemento de una lista:

```
ultimoElemento(X, [X]).
ultimoElemento(X, [A|B]) :- ultimoElemento(X, B).
```

- Hacer un procedimiento que agregue un elemento al final de una lista:

```
agregar([T], [], T).
agregar([H|L], [H|C], T) :- agregar(L, C, T).
```

Ejemplo: Procedimiento recursivo y cortes

- Procedimiento miembroDeLista/2: determina si un término aparece en una lista.

```
miembroDeLista(X, [X|_]).
miembroDeLista(X, [_|T]) :- miembroDeLista(X, T).
```

- Este procedimiento encuentra todas las ocurrencias. En muchas ocasiones es suficiente devolver verdadero una vez se encuentra la primera ocurrencia

```
miembroDeLista2(X, [X|_]) :- !.
miembroDeLista2(X, [_|T]) :- miembroDeLista2(X, T).
```

Demostración de validez de un procedimiento recursivo: ultimoElemento

- **Caso base**: Para la lista vacía el procedimiento falla (retorna falso). Esto es lo correcto, puesto que la lista vacía no tiene elementos.
- **Hipótesis**: El procedimiento es correcto para listas L.
- **Paso inductivo**: Tomamos la lista [H|L].
  - Si la lista [H|L] tiene un solo elemento, la primera regla unifica este elemento a la X y termina.
  - Si la lista [H|L] tiene más de un elemento, se llama recursivamente a el procedimiento con la lista L. Por hipótesis, esta llamada termina y devuelve el valor correcto.
- **Dominio bien fundado**.

Desarrollo descendente (Top-Down)

- Una forma de aproximación a la solución de un problema de programación
  - Se plantean los objetivos de más alto nivel y como resultado pueden aparecen nuevas tareas, las cuales a su vez pueden descomponerse.
- También se le conoce como *refinamiento progresivo*.

Ejemplo: Solución Top-Down

- Se tiene una lista con las notas del grupo. Se quiere saber el promedio.
- Para calcular el promedio se necesita la suma y el número de estudiantes.

```
mediaNotas(Lista, Media) :- longitudLista(Lista, Long),
                            sumaLista(Lista, Sum),
                            Media is Sum/Long.
```

Ejemplo: Continuación

- Se resuelven cada uno de los subproblemas

```
longitudLista([], 0).
longitudLista([_|Cola], Longitud) :-
      longitudLista(Cola, Longcola),
      Longitud is 1+Longcola.

sumaLista([], 0).
sumaLista([H|Cola], Total) :-
      sumaLista(Cola, Subtotal),
      Total is H+Subtotal.
```