Programación Elemental con Bases de Datos y Programación

- II.1 El lenguaje Prolog.
- II.2 Programación con bases de datos.
- II.3 Aspectos a tener en cuenta en la programación en Prolog
- II.4 Programación recursiva: Aritmética de los números naturales.
- II.5 Aritmética extra-lógica de Prolog.
- II.6 Predicados de entrada y salida en Prolog

El lenguaje Prolog

- Mecanización del método de resolución SLD
  - fijar la regla de selección de átomos del objetivo
  - fijar la regla de búsqueda de cláusulas en la base de conocimiento
  - incorporar backtracking para salir de las derivaciones fallidas
- Aumento de la expresividad del lenguaje incorporando facilidades
  - extralógicas (manipulación de expresiones aritméticas)
  - metalógicas (análisis y manipulación de elementos del lenguaje)
- Aprovechamiento de las capacidades de la máquina
  - gestión de entradas y salidas

Programación con bases de datos.

- Relación de la Programación Lógica con las Bases de Datos Relacionales

Dada una colección de dominios finitos $D_1,...,D_n$,

una **base de datos relacional** sobre estos dominios consistirá en un cierto número de relaciones

$$R \subseteq D_{i_1} \times \cdots \times D_{i_p}$$

definidas sobre algunos de los dominios.

Ejemplo: dados los dominios: *HOMBRE*, *MUJER*, *PERSONA*, se pueden definir las relaciones

$$PADRE \subseteq HOMBRE \times PERSONA$$

$$MADRE \subseteq MUJER \times PERSONA$$

$$MARIDO \subseteq HOMBRE \times MUJER$$

cuya descripción se suele dar de forma extensiva mediante tablas.

Programación Lógica

- Puede usarse para:
  - la **representación** de bases de datos relacionales, representado en la base de conocimientos de un programa las relaciones de forma:
    - explícita, mediante secuencias de hechos
    - implícita, mediante secuencia de reglas
  - la **consulta** mediante el planteamiento de objetivos que extraen información contenida en la base de datos.

Representación de dominios

- **Predicados monarios** para definir los tipos:

```
mujer(agar).
mujer(sara).
mujer(rebeca).
mujer(milka).
mujer(yiska).
```

- y llamar a estos predicados donde se quiera limitar los valores de alguna variable o controlar la consistencia de los datos.

```
tia(T,S) :-    progenitor(P,S),
               hermanos(T,P),
               mujer(T).
```

Representación de dominios

- Utilizar **estructuras de datos**:

```
dia(3,feb)   fecha(2,nov,1993)
```

y emplear el nombre de la estructura como patrón

```
aniversario(P,dia(D,M)):-
    nacimiento(P,fecha(D,M,A))
```

Representación de dominios

- Utilizar una **combinación** de las soluciones anteriores para la definición del dominio.

```
nat(cero).
nat(s(N)):- nat(N).

suma(cero,Y,Y) :- nat(Y).
suma(s(X),Y,s(Z)) :- suma(X,Y,Z).

par(cero).
par(s(s(N)):-par(N).
```

Representación de relaciones

- Extensional
  - Enumerando los elementos de la relación

```
padre(juan,ana).
padre(maria,luis).
padre(pedro,jose).
```

- Intensional
  - Mediante reglas que expresen la relación

```
abuelo(A,N):-
      padre(A,X),
      progenitor(X,N).
```

Relaciones reflexivas

Sin restricción de tipo	Con restricción de tipo
`r(a,b).`	`r(a,b).`
`...`	`...`
`r(X,X).`	`r(X,X):-t(X).`

Ejemplo:

```
mismaEdad(juan,antonio).
...
mismaEdad(X,X):-persona(X).
```

Relaciones simétricas

Como clausura simétrica de una relación asimétrica

Divergente	Convergente
`r(a,b).`	`r'(a,b).`
`...`	`...`
`r(X,Y):-r(Y,X).`	`r(X,Y):-r'(X,Y).`
	`r(X,Y):-r'(Y,X).`

Ejemplo mal

```
casados(X,Y):-casados(Y,X).
```

Ejemplo bien

```
casados(X,Y):-esposa(X,Y).
casados(X,Y):-esposa(Y,X).
```

ejc03.pl ejc031.pl

Relaciones transitivas

- Como clausura transitiva de una relación intransitiva

Divergente

```
r(a,b).
...
r(X,Y):- r(X,Z),
         r(Z,Y).
```

Convergente

```
r(a,b).
...
rt(X,Y):- r(X,Y).
rt(X,Y):- r(X,Z),
          rt(Z,Y).
```

Ejemplo:

```
arco(a,b).
arco(b,c).
...
```

Mal

```
cm(X,Y):- arco(X,Y).
cm(X,Y):- cm(X,Z),
          cm(Z,Y).
```

ejc04.pl

Bien

```
cm(X,Y):- arco(X,Y).
cm(X,Y):- arco(X,Z),
          cm(Z,Y).
```

ejc041.pl

Preorden

- **C**omo clausura reflexiva y transitiva de una intransitiva

Ejemplo: (Representación de un grafo orientado)

```
arco(a,b).
...
% clausura reflexiva
cm(X,X).
% clausura transitiva
cm(X,Y):-    arco(X,Z),
             cm(Z,Y).
```

ejc05.pl

Relación de equivalencia

- Como clausura reflexiva y transitiva de la clausura simétrica de una relación asimétrica)

Ejemplo: (Representación de un grafo no orientado)

```
arco(a,b).
...
% clausura simétrica
eje(X,Y):- arco(X,Y).
eje(X,Y):- arco(Y,X).
% clausura reflexiva
cm(X,X).
% clausura transitiva
cm(X,Y):- eje(X,Z),
          cm(Z,Y).
```

ejc06.pl

Algebra de relaciones. Unión

Esquema:

```
r(X):-r1(X).
r(X):-r2(X).
```

Ejemplo:

```
progenitor(P,H):-padre(P,H).
progenitor(P,H):-madre(P,H).
```

Algebra de relaciones. Interesección

Esquema:

```
r(X):-r1(X),
      r2(X).
```

Ejemplo:

```
votante(X):-  residente(X),
              mayorDeEdad(X).
```

Algebra de relaciones. Producto cartesiano

Esquema:

```
r(X,Y):-  r1(X),
          r2(Y).
```

Ejemplo:

```
pareja(X,Y):- hombre(X),
              mujer(Y).
```

Algebra de relaciones. Diferencia

Esquema:

```
r(X):-    r1(X),
          not r2(X).
```

Ejemplo:

```
contribuyente(X,Y):-
   residente(X),
   not extranjero(X).
```

Algebra de relaciones. Proyección

Esquema:

```
r(X):-    r1(X,Y).
```

Ejemplo:

```
espadre(X):- padre(X,Y).
```

Algebra de relaciones. Selección

Esquema:

```
r(X):-    r1(X,Y),
          cond(Y).
```

Ejemplo:

```
adolescente(X):- edad(X,E),
                 E < 18.
```

Aspectos a tener en cuenta. Orden de las reglas

```
padre(juan,antonio).
padre(juan,luis).
padre(juan,maria).
padre(maria,jose).

?- padre(X,Y).

{X/juan, Y/antonio}
{X/juan, Y/luis}
{X/juan, Y/maria}
{X/maria, Y/jose}
```

```
padre(maria,jose).
padre(juan,maria).
padre(juan,luis).
padre(juan,antonio).

?- padre(X,Y).

{X/maria, Y/jose}
{X/juan, Y/maria}
{X/juan, Y/luis}
{X/juan, Y/antonio}
```

Aspectos a tener en cuenta. Orden de las fórmulas

- El orden de aparición de los predicados en los cuerpos de las cláusulas puede afectar a:

1.- El orden en que se generan las soluciones:

2.- El número de cálculos que se deben realizar:

3 - La terminación de los cómputos:

`ejc07.pl`

Aspectos a tener en cuenta. Modos de uso

- El uso influye en la forma de definir el predicado

Ejemplo:

```
abuelo(X,Y):- progenitor(X,Z),
              progenitor(Z,Y).
```

es más apropiada para objetivos como `abuelo(juan,X).`

```
abuelo(X,Y):- progenitor(Z,Y),
              progenitor(X,Z).
```

es más apropiada para objetivos como `abuelo(X,jose).`

- En ocasiones, es mejor definir predicados distintos: abuelo/2 y nieto/2.

Modos de uso

```
% nieto(+,-)      (+,?)
nieto(X,Y):-progenitor(X,Z),
            progenitor(Z,Y).

% abuelo(-,+)       (?,+)
abuelo(X,Y):-  progenitor(Z,Y),
               progenitor(X,Z).
```

Definición de predicados

- Influye en:
  - Tamaño del espacio de búsqueda
  - La aparición de ciclos y ramas infinitas
  - La aparición de soluciones redundantes

Tamaño de la búsqueda

```
europeo(Persona):-      pais(Pais),
                        nacido(Persona,Pais),
                        enEuropa(Pais).
```

Aparición de ciclos y ramas infinitas

```
casados(X,Y):- casados(Y,X).

padre(X,Y):- hijo(Y,X).
hijo(X,Y):- padre(Y,X).

antepasado(X,Y):- progenitor(X,Y).
antepasado(X,Y):- antepasado(X,Z),
                  progenitor(Z,Y).
```

Soluciones redundantes

```
minimo(X,Y,X):- menorigual(X,Y).
minimo(X,Y,Y):- menorigual(Y,X).

?- minimo(3,3,M).
```

Programación recursiva

- Aritmética natural

  `Nat ::= Cero | Suc(Nat)`

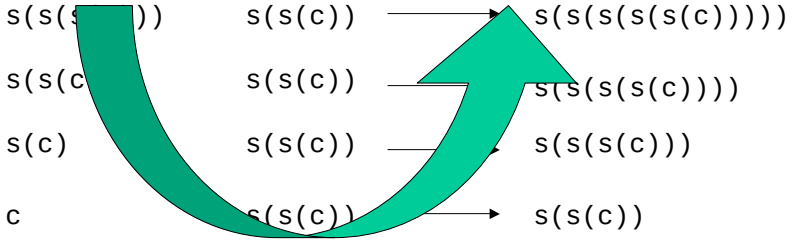
- A través de funtores. Dominio de Def.

```
c          0     nat(c).
s(c)       1     nat(s(N)):- nat(N).
s(s(c))    2
...        ..        ejc08.pl
```

Aritmética natural

Operaciones

Suma.




```
suma(c,M,M)       :- nat(M).
suma(s(N),M,s(X)) :- suma(N,M,X).
```

Tabla de comportamiento

- suma(X,Y,Z)

Uso	Comportamiento
(+,+,+)	Test. Comprueba si X+Y=Z
(+,+,-)	Generador único. Genera la suma Z = X+Y
(+,-,+)	Generador único. Genera la resta Y = Z-X
(-,+,+)	Generador único. Genera la resta X = Z-Y
(-,-,+)	Generador acotado. Genera todos los valores X e Y tal que Z = X+Y
(-,+,-)	Generador no acotado. Genera los naturales en X y la suma de X e Y en Z
(-,-,-)	Generador anómalo

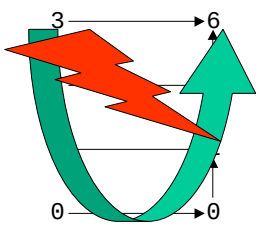
Aritmética natural

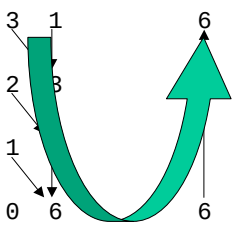
```
par                 mcd
impar               for
enPares             forStep
prod
menor
menorigual
min
factorial
```

Acumuladores

```
fact(X,Z)  :-    factA(X,s(c),Z).

factA(c,A,A)      :-    nat(A).
factA(s(X),A,Z)   :-    prod(s(X),A,NA),
                        factA(X,NA,Z).
```





Aritmética extralógica

X + Y	suma
X - Y	resta
X * Y	producto
X / Y	división
X // Y	división entera
-X	menos unario
X mod Y	módulo
abs(X)	valor absoluto
sqrt(X)	raíz cuadrada
exp(X), X**Y	e elevado a X, X elevado a Y
log(X), log10(X)	logaritmos neperianos y base 10
sin(X), cos(X), tan(X)	seno, coseno y tangente
asin(X), acos(X), atan(X)	arcoseno, arcocoseno y arcotangente
integer(X)	redondea a entero
float(X)	convierte a float
round(X),truncate(X), ceiling(X),floor(X)	redondeos
random(N)	número aleatorio 0 <= r < N

Aritmética extralógica

- Una expresión puede evaluarse si no contiene variables libres.
- Las expresiones anteriores, al evaluarse devuelven un valor numérico.
- Si el resultado de una expresión es erróneo, el predicado que la evalúa produce error.

Predicado is

```
X   is   E
```

- Evalúa la expresión E.
- Unifica el resultado con X (X puede ser número o variable)
  - Si unifican y X libre => éxito
    - » (X se instancia al valor).
  - Si unifican y X instanciada => éxito
  - Si no unifican => fallo
  - Si hay error y X libre => ERROR
  - Si hay error y X instanciada => ERROR

Ejemplo de is

`X is 2+5`
éxito `{X/7}`

`7 is 4+3`
éxito

`4+5 is 6+3`
fallo (la parte izquierda no se evalúa)

`X is a+1`
ERROR

`X is 3+Y` (con Y libre)
ERROR

`X is 3+Y` (con `{Y/5}`)
éxito `{X/8}`

Predicado is

`is` NO ES ASIGNACION

`X is X+1` (con `{X/4}`)

fallo

`X is X+1` (con X libre)

ERROR

`X is a, Y is X+1, Z is 5+9`

ERROR

`is` NO SE RESATISFACE NUNCA

Operadores relacionales

```
E1 >    E2      mayor
E1 >=   E2      mayor o igual
E1 <    E2      menor
E1 =<   E2      menor o igual
E1 =:=  E2      igual
E1 =\=  E2      distinto
```

- Se evalúan los dos argumentos y se compara los resultados.

Ejemplos

`3+7 > 2+5`

éxito

`2+4 =:= 3+X`

si `{X/3}` => éxito

si `X` está instanciada a un valor <>3 => fallo.

si `X` está libre => ERROR.

`3+1 =:= a+2`

ERROR

Revisión programas lógicos

`min`

`mcd`

`muma`

`factorial` (sin y con acumuladores)

`fibonacci` (sin y con acumuladores)

`for`

`forStep`

Menus

```
%% entradas
   entrada(rabanos, 20).
   entrada(pate, 90).
   entrada(ensalada, 40).
   entrada(sopa,35).
%% primer plato (de carne)
   carne(ternera, 130).
   carne(pollo, 100).
   carne(conejo, 120).
%% primer plato (de pescado)
   pescado(trucha, 90).
   pescado(salmon, 150).
   pescado(merluza, 95).
%% postre
   postre(flan, 80).
   postre(cuajada, 60).
   postre(fruta, 50).
```

```
"calmenu(E,P,T,C)
   C son las calorías
   del menú con entrada E,
   primer plato P y postre T"

calmenu(E,P,T,C) :-   entrada(E,CE),
                      pescado(P,CP),
                      postre(T,CT),
                      C is CE+CP+CT.
calmenu(E,P,T,C) :-   entrada(E,CE),
                      carne(P,CP),
                      postre(T,CT),
                      C is CE+CP+CT.
```

Cuestiones sobre menu

- ¿Qué menús tiene menos de 150 calorías?.
  `?-calmenu(E,P,T,C),C < 150.`
- ¿Y exactamente 200?.
  `?-calmenu(E,P,T,200).`
- ¿Qué menús tienen menos de 210 siendo el postre el que aporta más de 70?
  `?-calmenu(E,P,T,C),C<210,postre(T,CP),CP>70.`
- ¿Cuantas calorías tienen los menús que llevan ternera?
  `?-calmenu(E,ternera,T,C).`
- ¿Cuantas calorías tiene los menús que llevan carne?
  `?-calmenu(E,P,T,C),carne(P,CP).`

Rompecabezas

```
   C A M A
+  V I N O
---------
 V I C I O
```

```
     1   1
   5 6 2 9
+  3 8 0 7
---------
   9 4 3 6
```

```
Nac Ac
    X
    Y
---
    Z
```

```
num(N) :-for(0,9,N).
suma(Ac,X,Y,Z,Nac) :-   N is Ac+X+Y,
                        Z is N mod 10,
                        Nac is N // 10
```

Rompecabezas

```
     Y Z T 0
     C A M A
+    V I N O
-----------
   V I C I O
```

```
solucion(C,A,M,V,I,N,O):-
  num(A),num(O), A=\=O,
  suma(0,A,O,O,T),
  num(M), M=\=A, M=\=O,
  num(N), N=\=M, N=\=A, N=\=O,
  suma(T,M,N,I,Z), I=\=N, I=\=M, I=\=A, I=\=O,
  suma(Z,A,I,C,Y), C=\=I, C=\=N, C=\=M, C=\=A, C=\=O,
  num(V), V=\=C, V=\=I, V=\=N, V=\=M, V=\=A, V=\=O,
  suma(Y,C,V,I,V).
```

Ejercicios

```
    C A M A
    C A S A
+   M E S A
  V I C I O
```

```
    A  B  C - 15
    D  E  F - 15
    G  H  I - 15
  / |  |  | \
15 15 15 15 15
```

Entrada/Salida

Entrada

`read(X)`

X puede ser cualquier término o variable.

- Lee un término (Hay que terminarlo con un punto)
- Lo intenta unificar con X
  puede dar fallo o éxito
- Su comportamiento está fuera del modelo de Prolog.

Ejemplos

```
        ?- read(X)
juan.
        X = juan.
        yes.

        ?- read(suma(X,Y)).
suma(3,9).
        X = 3
        Y = 9
        yes
        ?-read(X),Y is X.
3+4.
        X = 3+4
        Y = 7.
        yes
```

Entrada/Salida

Salida

`write(X)`

- Escribe el término X en el dispositivo de salida.

```
    ?-X is 4+2, write(X).
    6
    X=6
    yes
    ?-read(X), Y is X,
     write('El valor de la variables leida es '),
     write(Y).
3+4.
    El valor de la variable leida es 7
    X = 3 + 4
    Y = 7
    yes
```

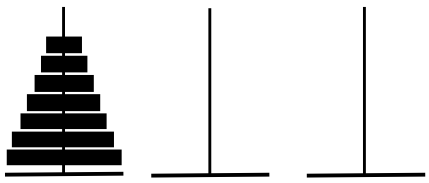
Entrada/Salida

`nl`

- escribe un salto de línea en el dispositivo de salida.

• Un evaluador de expresiones en Prolog

```
ev :- write('Escriba la expresion y punto'),
      nl,
      read(X), Y is X,
      write('El valor de '),write(X),
      write(' es '),write(Y).
?- ev.
Escriba la expresion y punto.
5+6.
El valor de 5+6 es 11
yes
```



```
%% hanoi(Discos,Origen,Ayuda,Destino).
hanoi(0,I,F,A).
hanoi(N,I,F,A):-     N > 0,
                     N1 is N - 1,
                     hanoi(N1,I,A,F),
                     write(I),write(' -> '),write(F),
                     nl,
                     hanoi(N1,A,F,I).

?- hanoi(3,pri,seg,ter).
```