



UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA

DEPARTAMENTO DE
INFORMÁTICA Y ANÁLISIS NUMÉRICO



PROGRAMACIÓN DECLARATIVA

INGENIERÍA INFORMÁTICA

CUARTO CURSO

PRIMER CUATRIMESTRE

Tema 11.- Reevaluación y el “corte”



Primera
parte:
Scheme

Tema 1.- Introducción al Lenguaje Scheme

Tema 2.- Expresiones y Funciones

Tema 3.- Predicados y sentencias condicionales

Tema 4.- Iteración y Recursión

Tema 5.- Tipos de Datos Compuestos

Tema 6.- Abstracción de Datos

Tema 7.- Lectura y Escritura

Segunda
parte: Prolog

Tema 8.- Introducción al Lenguaje Prolog

Tema 9.- Elementos Básicos de Prolog

Tema 10.- Listas

Tema 11.- **Reevaluación y el “corte”**

Tema 12.- Entrada y Salida

Segunda parte: Prolog

Tema 8.- Introducción al Lenguaje Prolog

Tema 9.- Elementos Básicos de Prolog

Tema 10.- Listas

Tema 11.- Reevaluación y el “corte”

Tema 12.- Entrada y Salida

Índice

1. Generación de múltiples soluciones
2. Predicados de control de ejecución

Índice

1. **Generación de múltiples soluciones**
2. Predicados de control de ejecución

1. Generación de múltiples soluciones

- Uso del punto coma
- *bagof*
- *setof*
- *findall*

1. Generación de múltiples soluciones

- **Uso del punto coma**
- *bagof*
- *setof*
- *findall*

1. Generación de múltiples soluciones

- **Uso del punto coma**

- Al hacer una pregunta,
 - el intérprete de Prolog genera la **primera solución** que encuentra.
- Si se desean **obtener más soluciones**
 - entonces se debe teclear **punto y coma “;”** tantas veces como se desee.

1. Generación de múltiples soluciones

- **Uso del punto coma**

- **Ejemplo 1: meses**

mes(enero,31).

mes(febrero,28).

mes(marzo,31).

mes(abril,30).

mes(mayo,31).

mes(junio,30).

mes(julio,31).

mes(agosto,31).

mes(septiembre,30).

mes(octubre,31).

mes(noviembre,30).

mes(diciembre,31).

1. Generación de múltiples soluciones

- **Uso del punto coma**

- **Ejemplo 1: meses**

- Meses que tienen 31 días

?- *mes(M,31).*

M = enero ;

M = marzo ;

M = mayo ;

M = julio ;

M = agosto ;

M = octubre ;

M = diciembre ;

No

Se teclea “;”

1. Generación de múltiples soluciones

- **Uso del punto coma**
 - **Ejemplo 2: personas**

persona(patricia,9,femenino).

persona(laura,9,femenino).

persona(juan,9,masculino).

persona(teresa,8,femenino).

persona(pedro,8,masculino).

persona(laura,8,femenino).

1. Generación de múltiples soluciones

- **Uso del punto coma**

- **Ejemplo 2: personas**

- Niñas con 9 años

?- *persona(P,9,femenino).*

P = patricia ;

P = laura ;

No



Se teclea “;”

1. Generación de múltiples soluciones

- Uso del punto coma
- *bagof*
- *setof*
- *findall*

1. Generación de múltiples soluciones

- *bagof*

- Sintaxis

- ?- *bagof* (*Plantilla*, *Objetivo*, *Lista*)

- Significado

- **Plantilla**

- Indica el **formato** para guardar en la lista los valores que hagan verdadero el “objetivo”.

- **Objetivo**

- Expresión lógica.

- **Lista**

- Usa la **plantilla** para almacenar los **valores** que hacen verdadero el “objetivo”.

1. Generación de múltiples soluciones

- *bagof*

- **Observación**

- Pueden aparecer **valores repetidos** en la lista de resultados.

1. Generación de múltiples soluciones

- *bagof*

- Ejemplo 1: meses

- Lista de meses que tienen 31 días

?- *bagof*(*M*,*mes*(*M*,31),*L*).

L = [*enero*, *marzo*, *mayo*, *julio*, *agosto*, *octubre*,
diciembre]

?- *bagof*(*M/31*,*mes*(*M*,31),*L*).

L = [*enero/31*, *marzo/31*, *mayo/31*, *julio/31*,
agosto/31, *octubre/31*, *diciembre/31*]

1. Generación de múltiples soluciones

- *bagof*

- Ejemplo 1: meses

- Lista de meses y días

?- *bagof*(*M/D*,*mes*(*M*,*D*),*L*).

L = [*enero/31*, *febrero/28*, *marzo/31*, *abril/30*,
mayo/31, *junio/30*, *julio/31*, *agosto/31*, ... /... |...]

?- *bagof*((*M*,*D*),*mes*(*M*,*D*),*L*).

L = [(*enero*, 31), (*febrero*, 28), (*marzo*, 31), (*abril*, 30), (*mayo*, 31), (*junio*, 30), (*julio*, 31), (*agosto*, 31), (... , ...) |...].

1. Generación de múltiples soluciones

- *bagof*

- Ejemplo 1: meses

- Lista de **meses** agrupados según los **días** que tienen

?- *bagof*(*M*,*mes*(*M*,*D*),*L*).

D = 28,

L = [*febrero*] ;

D = 30,

L = [*abril, junio, septiembre, noviembre*] ;

D = 31,

L = [*enero, marzo, mayo, julio, agosto, octubre, diciembre*].

Se teclea “;”

1. Generación de múltiples soluciones

- *bagof*

- Observación

- Se puede usar el **operador existencial** “ \wedge ” para que **no** tenga en cuenta los valores de las variables **libres** o **no enlazadas**.

1. Generación de múltiples soluciones

- *bagof*

- Ejemplo 1: meses

- Lista de meses **sin** tener en cuenta los **días** que tienen

?- *bagof*(M,D[^]mes(M,D),L).

L = [enero, marzo, mayo, julio, agosto, septiembre | ...]

- **Nota:** el uso de la variable anónima no genera todas las soluciones

?- *bagof*(M,mes(M,_),L).

L = [febrero] .

1. Generación de múltiples soluciones

- *bagof*

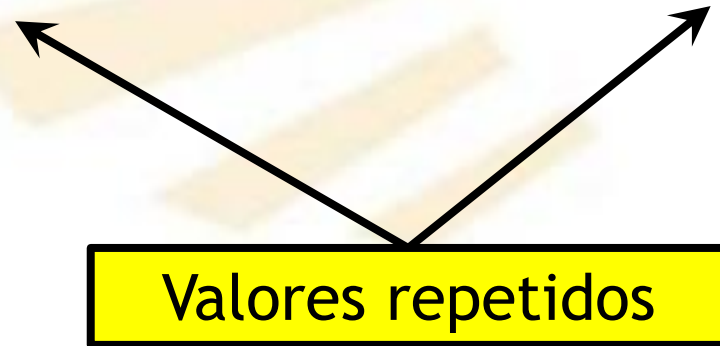
- Ejemplo 2: personas

buscar_personas_1(L):-

bagof(P, E^S^persona(P,E,S), L).

? *buscar_personas_1(L).*

L = [patricia, laura, juan, teresa, pedro, laura]



1. Generación de múltiples soluciones

- *bagof*

- Ejemplo 2: personas

buscar_por_edad(E,L):-

bagof(P,S^persona(P,E,S),L).

?- *buscar_por_edad(9,L).*

L = [patricia, laura, juan].

?- *buscar_por_edad(8,L).*

L = [teresa, pedro, laura].

1. Generación de múltiples soluciones

- *bagof*

- Ejemplo 2: personas

buscar_por_edad(E,L):-

bagof(P,S^persona(P,E,S),L).

?- *buscar_por_edad(E,L).*

E = 8,

L = [teresa, pedro, laura] ;

E = 9,

L = [patricia, laura, juan].

Se teclea “;”

1. Generación de múltiples soluciones

- *bagof*

- Ejemplo 2: personas

buscar_por_sexo(S,L):-

bagof(P,E^persona(P,E,S),L).

? *buscar_por_sexo(S,L):-*

Se teclea “;”

S = femenino,

L = [patricia, laura, teresa, laura] ;

S = masculino,

L = [juan, pedro].

Valores repetidos

1. Generación de múltiples soluciones

- *bagof*

- Ejemplo 2: personas

buscar_por_sexo(S,L):-

bagof(P,E^persona(P,E,S),L).

?- *buscar_por_sexo(femenino,L).*

L = [patricia, laura, teresa, laura].

?- *buscar_por_sexo(masculino,L).*

L = [juan, pedro].

1. Generación de múltiples soluciones

- *bagof*
 - Ejemplo 2: personas
 - Niñas de 9 años

?- *bagof*(*P*,*persona*(*P*,9,*femenino*),*L*).

L = [*patricia*, *laura*].

1. Generación de múltiples soluciones

- Uso del punto coma
- *bagof*
- *setof*
- *findall*

1. Generación de múltiples soluciones

- *setof*

- Sintaxis

- ? *setof* (*Plantilla, Objetivo, Lista*)

- Significado

- **Plantilla:**

- Indica el **formato** para guardar en la lista los valores que hagan verdadero el “objetivo”.

- **Objetivo:**

- Expresión lógica.

- **Lista:**

- Usa la **plantilla** para almacenar **de forma ordenada** los valores que hacen verdadero el “objetivo”.

- No** hay repeticiones

1. Generación de múltiples soluciones

- *setof*

- Ejemplo 1: meses

- Lista ordenada alfabéticamente de los meses con 31 días

?- *setof*(*M*,*mes*(*M*,31),*L*).

L = [*agosto*, *diciembre*, *enero*, *julio*, *marzo*,
mayo, *octubre*].

?- *setof*(*M*/*31*,*mes*(*M*,31),*L*).

L = [*agosto/31*, *diciembre/31*, *enero/31*,
julio/31, *marzo/31*, *mayo/31*, *octubre/31*].

1. Generación de múltiples soluciones

- *setof*

- Ejemplo 1: meses

- Lista ordenada alfabéticamente de los meses **sin** tener en cuenta los **días**

?- *setof*(*M*, *D*^*mes*(*M*, *D*), *L*).

L = [*abril*, *agosto*, *diciembre*, *enero*, *febrero*,
julio, *junio*, *marzo*, *mayo* | ...].

1. Generación de múltiples soluciones

- *setof*

- **Ejemplo 1: meses**

- **Lista ordenada alfabéticamente de los meses con sus días**

?- *setof*(*M/D*,*mes*(*M*,*D*),*L*).

L = [*abril/30*, *agosto/31*, *diciembre/31*, *enero/31*,
febrero/28, *julio/31*, *junio/30*, *marzo/31*, ... / ... | ...].

1. Generación de múltiples soluciones

- *setof*

- **Ejemplo 2: personas**

- Lista de personas ordenadas por edades **sin** tener en cuenta el **sexo**

buscar_ordenado_por_edad(E,L):-

setof(P,S^persona(P,E,S),L).

?- *buscar_ordenado_por_edad(E,L).*

E = 8,

L = [laura, pedro, teresa] ; ←

Se teclea “;”

E = 9,

L = [juan, laura, patricia].

1. Generación de múltiples soluciones

- *setof*

- **Ejemplo 2: personas**

- Lista de personas con 9 años ordenadas alfabéticamente

buscar_ordenado_por_edad(E,L):-

setof(P,S^persona(P,E,S),L).

?- *buscar_ordenado_por_edad(9,L).*

L = [juan, laura, patricia].

1. Generación de múltiples soluciones

- *setof*

- **Ejemplo 2: personas**

- **Lista de niñas con 9 años ordenadas alfabéticamente**

? *setof*(**P**, *persona*(**P**, 9, *femenino*), **L**).

L = [*laura*, *patricia*].

1. Generación de múltiples soluciones

- *setof*

- **Ejemplo 2: personas**

- Lista de personas ordenadas por sexo **sin** tener en cuenta la **edad**

buscar_ordenado_por_sexo(S,L):-

setof(P,E^persona(P,E,S),L).

?- *buscar_ordenado_por_sexo(S,L).*

S = femenino,

L = [laura, patricia, teresa] ;

S = masculino,

L = [juan, pedro].

Se tecldea “;”

1. Generación de múltiples soluciones

- Uso del punto coma
- *bagof*
- *setof*
- *findall*

1. Generación de múltiples soluciones

- *findall*

- Sintaxis

?- *findall* (Plantilla, Objetivo, Lista)

- Significado

- Su funcionamiento es similar a *bagof* cuando enlaza todas las variables libres con el operador existencial “^”.
- Si no hay soluciones
 - *bagof* falla.
 - *findall* genera una lista vacía.

1. Generación de múltiples soluciones

- *findall*

- **Ejemplo**

- Lista de meses **sin** tener en cuenta los **días**

?- *findall*(**M**,mes(**M**,**D**),L).

L = [enero, febrero, marzo, abril, mayo, junio, julio, agosto, septiembre|...];

- Uso equivalente de *bagof*

?- *bagof*(**M**,**D**^mes(**M**,**D**),L).

L = [enero, febrero, marzo, abril, mayo, junio, julio, agosto, septiembre|...].

1. Generación de múltiples soluciones

- *findall*

- **Ejemplo: personas**

- Lista de personas

?- *findall*(*P*,*persona*(*P*,*E*,*S*),*L*).

L = [*patricia*, *laura*, *juan*, *teresa*, *pedro*, *laura*].

?- *findall*(*P*,*persona*(*P*,_,*S*),*L*).

L = [*patricia*, *laura*, *juan*, *teresa*, *pedro*, *laura*].

?- *findall*(*P*,*persona*(*P*,*E*,_),*L*).

L = [*patricia*, *laura*, *juan*, *teresa*, *pedro*, *laura*].

?- *findall*(*P*,*persona*(*P*,_,_),*L*).

L = [*patricia*, *laura*, *juan*, *teresa*, *pedro*, *laura*].

1. Generación de múltiples soluciones

- *findall*

- **Ejemplo: personas**

- Lista de personas femeninas **sin** tener en cuenta la **edad**

buscar_por_sexo_2(S,L):-

findall(P,persona(P,_,S),L).

?- *buscar_por_sexo_2(femenino,L).*

L = [patricia, laura, teresa, laura].

1. Generación de múltiples soluciones

- *findall*

- **Ejemplo: personas**

- Lista de personas de una edad determinada

buscar_por_edad_2(E,L):-

findall(P,persona(P,E,_),L).

?- *buscar_por_edad(9,L).*

L = [patricia, laura, juan].

1. Generación de múltiples soluciones

- *findall*

- **Ejemplo: personas**

- Número de personas de una edad determinada
contar([],0).

contar([_ | Cola],N):-

contar(Cola,N1),

N is N1+1.

numero_personas_por_edad(E,N):-

findall(P,persona(P,E,_),L),

contar(L,N).

? numero_personas_por_edad(9,N).

N = 3

Índice

1. Generación de múltiples soluciones
2. **Predicados de control de ejecución**

2. Predicados de control de ejecución

- *true*
- *fail*
- *repeat*
- El corte: !
- *call*
- El condicional: ->

2. Predicados de control de ejecución

- *true*
- *fail*
- *repeat*
- El corte: !
- *call*
- El condicional: ->

2. Predicados de control de ejecución

- *true*

- Sintaxis

true

- Significado

- Este predicado siempre tiene éxito.

- Ejemplo

- El hecho

edad(juan, 12).

es equivalente a

edad(juan, 12):- true.

2. Predicados de control de ejecución

- *true*
 - Observación
 - Se utiliza como elemento auxiliar del condicional “->”.

2. Predicados de control de ejecución

- *true*
- *fail*
- *repeat*
- El corte: !
- *call*
- El condicional: ->

2. Predicados de control de ejecución

- *fail*

- Sintaxis

fail

- Significado

- Este predicado siempre fracasa.

- Usos

- Repetición de un proceso

- ✓ *failure driven loop*

- ✓ Uso de *fail* con *repeat*

- Con *el corte !*:

- ✓ Provoca un **fallo inmediato** y termina una búsqueda que no va a tener éxito.

2. Predicados de control de ejecución

- *fail*

- *failure driven loop: esquema*

```
/* Bucle */
```

```
failure_driven_loop(Dato):-
```

```
    generador(Dato,Término),
```

```
    efecto_colateral(Término),
```

```
    fail.
```

```
/* Fin */
```

```
failure_driven_loop(Dato).
```

- Se repite el bucle hasta que *generador* no sea verdadero

2. Predicados de control de ejecución

- *fail*

- Ejemplo: *failure driven loop*

padre_de(juan, pepe).

padre_de(juan, luis).

padre_de(juan, alberto).

listado:-padre_de(juan,X), write(X), nl, *fail*.

listado.



generador



efecto colateral

2. Predicados de control de ejecución

- *fail*
 - Ejemplo: *failure driven loop*

```
/* “listado” no tiene argumentos */
```

```
?- listado.
```

```
pepe
```

```
luis
```

```
alberto
```

```
true.
```

2. Predicados de control de ejecución

- *true*
- *fail*
- *repeat*
- El corte: !
- *call*
- El condicional: ->

2. Predicados de control de ejecución

- *repeat*

- **Sintaxis**

repeat

- **Significado**

- Es usado para simular un bucle.
- Se repite el predicado en el que es usado hasta que tiene éxito.
- Se suele utilizar en procesos de lectura.

2. Predicados de control de ejecución

- *repeat*

- **Ejemplo**

- Lee hasta que se introduce el número 9

```
test :- repeat,
```

```
    write('Introduce un número -->'),
```

```
    read(X),
```

```
    (X==9).
```

```
?- test.
```

```
Introduce un número --> 3.
```

```
Introduce un número --> 9.
```

```
true.
```

2. Predicados de control de ejecución

- *repeat*

- Ejemplo

- Calcula el cuadrado de un número hasta que se lee el carácter de fin de fichero.

```
cuadrado(X,R):- R is X * X.
```

```
test :- repeat,
```

```
    write('Introduce un número -->'),
```

```
    read(Dato),
```

```
    nl,
```

```
    ( Dato == end_of_file;
```

```
        cuadrado(Dato, R), write(R), nl, fail
```

```
    ).
```


2. Predicados de control de ejecución

- *repeat*

- **Ejemplo**

- Calcula el cuadrado de un número hasta que se lee el carácter de fin de fichero.

?- *test*.

Introduce un número -->2.

4

Introduce un número -->3.

9

*Introduce un número --> **Control D***

true.

2. Predicados de control de ejecución

- *repeat*

- Ejemplo

- Estructura general

```
test:- repeat,  
      write('Introduce un dato -->'),  
      read(Dato),  
      ( Dato == end_of_file;  
        proceso(Dato), fail  
      ).
```

- Restricción

- Se requiere que el *proceso* siempre tenga éxito con el dato de entrada.

2. Predicados de control de ejecución

- *true*
- *fail*
- *repeat*
- **El corte: !**
- *call*
- El condicional: ->

2. Predicados de control de ejecución

- **El corte: !**
 - Sintaxis y significado
 - Comparación entre “not” y el corte “!”
 - Aplicaciones

2. Predicados de control de ejecución

- **El corte: !**
 - **Sintaxis y significado**
 - Comparación entre “not” y el corte “!”
 - Aplicaciones

2. Predicados de control de ejecución

- **El corte: !**

- **Sintaxis**

- !

- **Significado**

- Es un predicado especial que siempre es cierto.
 - No puede volver a ser evaluado.
 - El origen de ! es el símbolo matemático

$$(\exists!x) A(x)$$

que indica que existe un único x tal que $A(x)$.

2. Predicados de control de ejecución

- **El corte: !**
 - Sintaxis y significado
 - **Comparación entre “not” y el corte “!”**
 - Aplicaciones

2. Predicados de control de ejecución

- **El corte: !**
 - **Comparación entre “not” y el corte “!”**
 - Descripción
 - Ejemplos

2. Predicados de control de ejecución

- **El corte: !**
 - **Comparación entre “not” y el corte “!”**
 - **Descripción**
 - **Ejemplos**

2. Predicados de control de ejecución

- El corte: !

- Comparación entre “not” y el corte “!”

- Descripción

- Uso del predicado “not”: claridad semántica

$A:- B, C.$

$A:- \text{not}(B), D.$

- Uso del corte: eficiencia

$A:-B, !, C.$

$A:- D.$

- ✓ La segunda regla no necesita comprobar que B no se cumple.

2. Predicados de control de ejecución

- El corte: !

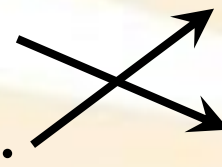
- Comparación entre “not” y el corte “!”

- Descripción

- “not” permite el intercambio de las reglas

$A:- B, C.$

$A:- \text{not}(B), D.$



$A:- \text{not}(B), D.$

$A:- B, C.$

- El corte no permite el intercambio de las reglas

$A:-B, !, C.$

$A:- D.$



$A:- D.$

$A:-B, !, C.$

✓ **Cambia** el significado de las reglas

2. Predicados de control de ejecución

- **El corte: !**
 - **Comparación entre “not” y el corte “!”**
 - Descripción
 - Ejemplos

2. Predicados de control de ejecución

- **El corte: !**
 - **Comparación entre “not” y el corte “!”**
 - Ejemplos
 1. Nota alfanumérica
 2. Factorial de un número
 3. Veces
 4. Deportistas

2. Predicados de control de ejecución

- **El corte: !**
 - **Comparación entre “not” y el corte “!”**
 - Ejemplos
 1. Nota alfanumérica
 2. Factorial de un número
 3. Veces
 4. Deportistas

2. Predicados de control de ejecución

- El corte: !
 - Comparación entre “not” y el corte “!”
 - Ejemplo 1.- Nota alfanumérica

nota(X,suspenso):- X < 5.

nota(X,aprobado):- 5 =< X, X < 7.

nota(X,notable):- 7 =< X, X < 9.

nota(X,sobresaliente):- 9 =< X, X < 10.

nota(X, matricula_honor):- X = 10.

2. Predicados de control de ejecución

- El corte: !
 - Comparación entre “not” y el corte “!”
 - Ejemplo 1.- Nota alfanumérica

/ Se utiliza el corte ! */*

No es necesario
comprobar si $X \geq 5$

nota_bis(X,suspenso):- X < 5, !.

nota_bis(X,aprobado):- X < 7, !. ←

nota_bis(X,notable):- X < 9, !.

nota_bis(X,sobresaliente):- X < 10, !.

nota_bis(10, matricula_honor).

2. Predicados de control de ejecución

- El corte: !

- Comparación entre “not” y el corte “!”

- Ejemplo 1.- Nota alfanumérica

?- *nota(5,C).*

C = aprobado .

?- *nota(10,C).*

C = matricula_honor.

?- *nota_bis(5,C).*

C = aprobado.

?- *nota_bis(10,C).*

C = matricula_honor.

2. Predicados de control de ejecución

- El corte: !

- Comparación entre “not” y el corte “!”

- Ejemplo 1.- Nota alfanumérica

- ✓ Observación:

- Un uso incorrecto puede generar soluciones **erróneas**

*? nota(3,sobresaliente).
false*

Error

*? nota_bis(3,sobresaliente).
true*

2. Predicados de control de ejecución

- **El corte: !**
 - **Comparación entre “not” y el corte “!”**
 - Ejemplo 1.- Nota alfanumérica
 - Se pueden intercambiar las reglas sin que cambie el significado del programa.

nota(X, matricula_honor):- X = 10.

nota(X, sobresaliente):- 9 =< X, X < 10.

nota(X, notable):- 7 =< X, X < 9.

nota(X, aprobado):- 5 =< X, X < 7.

nota(X, suspenso):- X < 5.

2. Predicados de control de ejecución

- El corte: !
 - Comparación entre “not” y el corte “!”
 - Ejemplo 1.- Nota alfanumérica
 - El corte **no** permite intercambiar las reglas

nota_bis(10, matricula_honor).

nota_bis(X,sobresaliente):- X < 10, !.

nota_bis(X,notable):- X < 9, !.

nota_bis(X,aprobado):- X < 7, !.

nota_bis(X,suspenso):- X < 5, !.

Error

2. Predicados de control de ejecución

- **El corte: !**

- **Comparación entre “not” y el corte “!”**

- **Ejemplos**

1. Nota alfanumérica

2. **Factorial de un número**

3. Veces

4. Deportistas

2. Predicados de control de ejecución

- El corte: !

- Comparación entre “not” y el corte “!”

- Ejemplo 2.- Factorial de un número

- Primera versión

- ✓ Se usa la negación *not*

- ✓ **No** controla si N tiene un valor negativo
factorial_1(0, 1).

factorial_1(N,R):- not(N = 0),

N1 is N - 1,

factorial_1(N1,R1),

*R is N * R1.*

2. Predicados de control de ejecución

- El corte: !
 - Comparación entre “not” y el corte “!”
 - Ejemplo 2.- Factorial de un número
 - Segunda versión
 - ✓ Se usa la negación *not*
 - ✓ Se controla si N tiene un valor negativo

```
factorial_2(N, 1):- N =< 0.
```

```
factorial_2(N,R):-
```

```
    not(N =< 0), /* equivalente a N > 0 */
```

```
    N1 is N - 1,
```

```
    factorial_2(N1,R1),
```

```
    R is N * R1.
```

2. Predicados de control de ejecución

- El corte: !
 - Comparación entre “not” y el corte “!”
 - Ejemplo 2.- Factorial de un número
 - Tercera versión
 - ✓ Se usa el **corte**
 - ✓ **No** se controla si N tiene un valor negativo

```
factorial_3(0,1):- !.  
factorial_3(N,R):-  
    N1 is N - 1,  
    factorial_3(N1,R1),  
    R is N * R1.
```


2. Predicados de control de ejecución

- **El corte: !**
 - **Comparación entre “not” y el corte “!”**
 - Ejemplo 2.- Factorial de un número
 - Cuarta versión
 - ✓ Se usa el **corte**
 - ✓ Se controla si N tiene un valor negativo

factorial_4(N, 1):- N =< 0, !.

*factorial_4(N,R):-
N1 is N - 1,
factorial_4(N1,R1),
R is N * R1.*

2. Predicados de control de ejecución

- **El corte: !**
 - **Comparación entre “not” y el corte “!”**
 - Ejemplos
 1. Nota alfanumérica
 2. Factorial de un número
 3. **Veces**
 4. Deportistas

2. Predicados de control de ejecución

- **El corte: !**
 - **Comparación entre “not” y el corte “!”**
 - Ejemplo 3.- Veces
 - Cuenta el número de veces que un elemento X aparece en una lista simple L.

2. Predicados de control de ejecución

- El corte: !
 - Comparación entre “not” y el corte “!”
 - Ejemplo 3.- Veces
 - Versión con “not”

veces(_,[],0).

veces(X,[X|Cola],N):- *veces*(X,Cola,N1),

N is N1 + 1.

veces(X,[Y|Cola],N):- **not**(X = Y),

veces(X,Cola,N).

2. Predicados de control de ejecución

- **El corte: !**
 - **Comparación entre “not” y el corte “!”**
 - Ejemplo 3.- Veces
 - Versión con “el corte !”

veces_bis(_,[],0).

veces_bis(X,[X|Cola],N):- !,

veces_bis(X,Cola,N1),

N is N1 + 1.

veces_bis(X,[_ | Cola],N):- *veces_bis*(X,Cola,N).

2. Predicados de control de ejecución

- **El corte: !**
 - **Comparación entre “not” y el corte “!”**
 - Ejemplos
 1. Nota alfanumérica
 2. Factorial de un número
 3. Veces
 4. **Deportistas**

2. Predicados de control de ejecución

- **El corte: !**
 - **Comparación entre “not” y el corte “!”**
 - Ejemplo 4.- Deportistas
 - Clases de deportivas
 - ✓ **luchador:** gana y pierde
 - ✓ **ganador:** siempre gana, es decir, gana y no pierde.
 - ✓ **deportista:** siempre pierde, es decir, pierde y no gana.

2. Predicados de control de ejecución

- El corte: !

- Comparación entre “not” y el corte “!”

- Ejemplo 4.- Deportistas

vence(luis, diego).

vence(ana, luis).

vence(ana, diego).

2. Predicados de control de ejecución

- El corte: !
 - Comparación entre “not” y el corte “!”
 - Ejemplo 4.- Deportistas
 - Versión con “not”

```
clase(P, luchador):- vence(P,_),  
                    vence(_,P).
```

```
clase(P, ganador):- vence(P,_),  
                    not(vence(_,P)).
```

```
clase(P, deportista):- vence(_,P),  
                       not(vence(P,_)).
```

2. Predicados de control de ejecución

- **El corte: !**
 - **Comparación entre “not” y el corte “!”**
 - Ejemplo 4.- Deportistas
 - Versión con “el corte !”

```
clase_bis(P, luchador):- vence(P,_),  
                           vence(_,P),  
                           !.
```

```
clase_bis(P, ganador):- vence(P,_),  
                           !.
```

```
clase_bis(P, deportista):- vence(_,P),  
                           !.
```

2. Predicados de control de ejecución

- **El corte: !**
 - Sintaxis y significado
 - Comparación entre “not” y el corte “!”
 - **Aplicaciones**

2. Predicados de control de ejecución

- **El corte: !**
 - **Aplicaciones**
 - Evita la búsqueda de soluciones alternativas inexistentes.
 - Confirma el uso de una regla
 - Provoca un fallo inmediato: corte y *fail*

2. Predicados de control de ejecución

- **El corte: !**
 - **Aplicaciones**
 - **Evita la búsqueda de soluciones alternativas inexistentes.**
 - **Confirma el uso de una regla**
 - **Provoca un fallo inmediato: corte y *fail***

2. Predicados de control de ejecución

- **El corte: !**
 - **Aplicaciones**
 - **Evita la búsqueda de soluciones alternativas inexistentes**
 - Si se sabe que la **solución es única** entonces impide la búsqueda de soluciones alternativas.

2. Predicados de control de ejecución

- **El corte: !**
 - **Evita la búsqueda de soluciones alternativas inexistentes**
 - **Ejemplo: función definida por partes**

$$f(x) = \begin{cases} 0 & x < 3 \\ 2 & 3 \leq x < 6 \\ 4 & 6 \leq x \end{cases}$$

2. Predicados de control de ejecución

- **El corte: !**
 - **Evita la búsqueda de soluciones alternativas inexistentes**
 - Ejemplo: función definida por partes
 - **Primera versión**

$$f(X,0) :- X < 3.$$

$$f(X,2) :- 3 \leq X, X < 6.$$

$$f(X,4) :- 6 \leq X.$$

2. Predicados de control de ejecución

- **El corte: !**
 - **Evita la búsqueda de soluciones alternativas inexistentes**
 - Ejemplo: función definida por partes
 - **Primera versión**

La pregunta

$$?- f(1,R), 2 < R.$$

false

hace intentos de búsqueda superfluos

2. Predicados de control de ejecución

- **El corte: !**
 - **Evita la búsqueda de soluciones alternativas inexistentes**
 - Ejemplo: función definida por partes
 - **Primera versión**

?- $f(1, R), 2 < R.$

✓ **Primer intento: primera regla**

$f(X, 0) :- X < 3.$

$1 < 3$ (verdadero)

(R toma el valor 0)

$2 < 0$

false

2. Predicados de control de ejecución

- **El corte: !**
 - **Evita la búsqueda de soluciones alternativas inexistentes**
 - Ejemplo: función definida por partes
 - **Primera versión**

?- $f(1, R), 2 < R.$

✓ **Segundo intento: segunda regla**

$f(X, 2) :- 3 \leq X, X < 6.$

$3 \leq 1, 1 < 6.$

false

2. Predicados de control de ejecución

- **El corte: !**
 - **Evita la búsqueda de soluciones alternativas inexistentes**
 - Ejemplo: función definida por partes
 - **Primera versión**

?- $f(1, R), 2 < R.$

✓ **Tercer intento: tercera regla**

$f(X, 4) :- 6 =< X.$

$6 =< 1.$

false

2. Predicados de control de ejecución

- **El corte: !**
 - **Evita la búsqueda de soluciones alternativas inexistentes**
 - Ejemplo: función definida por partes
 - **Segunda versión: usa “el corte !”**

$g(X,0) :- X < 3, !.$

$g(X,2) :- 3 \leq X, X < 6, !.$

$g(X,4) :- 6 \leq X$

2. Predicados de control de ejecución

- **El corte: !**
 - **Evita la búsqueda de soluciones alternativas inexistentes**
 - Ejemplo: función definida por partes
 - **Segunda versión: usa “el corte !”**

La pregunta

?- $g(1,R), 2 < R.$

false

y **termina**, porque solamente hace un intento de búsqueda.

2. Predicados de control de ejecución

- **El corte: !**
 - **Evita la búsqueda de soluciones alternativas inexistentes**

- Ejemplo: función definida por partes

- **Segunda versión: usa “el corte !”**

?- $g(1,R), 2 < R.$

Se prueba la primera regla

$g(X,0) :- X < 3, !.$

$1 < 3$ (verdadero)

R toma el valor 0 , se **alcanza** el corte !

(impide el retroceso).

$2 < 0$

false

2. Predicados de control de ejecución

- **El corte: !**
 - **Evita la búsqueda de soluciones alternativas inexistentes**
 - Ejemplo: función definida por partes
 - **Tercera versión: se usa el corte**

$h(X,0) :- X < 3, !.$

$h(X,2) :- X < 6, !.$

$h(_,4).$

El corte simplifica la escritura de las reglas

2. Predicados de control de ejecución

- **El corte: !**
 - **Evita la búsqueda de soluciones alternativas inexistentes**
 - Ejemplo: función definida por partes
 - **Tercera versión: se usa el corte**

La pregunta

?- $h(1,R), 2 < R.$

false

y **termina**, porque solamente hace un intento de búsqueda.

2. Predicados de control de ejecución

- El corte: !

- Evita la búsqueda de soluciones alternativas inexistentes

- Ejemplo: función definida por partes

- Tercera versión: se usa el corte

?- $h(1, R), 2 < R.$

Se prueba la primera regla

$h(X, 0) :- X < 3, !.$

$1 < 3$ (verdadero)

R toma el valor 0 , se alcanza el corte !
(impide el retroceso).

$2 < 0$

false

2. Predicados de control de ejecución

- **El corte: !**
 - **Aplicaciones**
 - Evita la búsqueda de soluciones alternativas inexistentes.
 - **Confirma el uso de una regla**
 - Provoca un fallo inmediato: corte y *fail*

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplos:
 1. Máximo
 2. Máximo común divisor
 3. Crear lista de enteros
 4. Elementos
 5. Extremos
 6. Suprimir
 7. Préstamo de libros

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplos:
 1. **Máximo**
 2. **Máximo común divisor**
 3. **Crear lista de enteros**
 4. **Elementos**
 5. **Extremos**
 6. **Suprimir**
 7. **Préstamo de libros**

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplo 1.- Máximo

max(X, Y, X):- X >= Y, !.

max(_, Y, Y).

?- max(3, 5, R).

R = 5.

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplos:
 1. Máximo
 2. **Máximo común divisor**
 3. Crear lista de enteros
 4. Elementos
 5. Extremos
 6. Suprimir
 7. Préstamo de libros

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplo 2.- Máximo común divisor

mcd(X,0,X):-!.

mcd(X,Y,M):- X >= Y, !,

X1 is X mod Y,

mcd(Y,X1,M).

mcd(X,Y,M):- mcd(Y,X,M).

?- max(12,18,R).

R =6.

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplos:
 1. Máximo
 2. Máximo común divisor
 - 3. Crear lista de enteros**
 4. Elementos
 5. Extremos
 6. Suprimir
 7. Préstamo de libros

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplo 3.- Crear lista de enteros
crear_enteros(0,[0]):-!.

*crear_enteros(N,[N|L]):- N1 is N-1,
crear_enteros(N1,L).*

?- crear_enteros(5,L).

L = [5,4,3,2,1,0]

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplos:
 1. Máximo
 2. Máximo común divisor
 3. Crear lista de enteros
 - 4. Elementos**
 5. Extremos
 6. Suprimir
 7. Préstamo de libros

2. Predicados de control de ejecución

- **El corte: !**

- **Confirma el uso de una regla**

- **Ejemplos 4.- Elementos**

- Extrae los elementos de una lista con sublistas
elementos([],[]).

- elementos([X | Cola],R):- elementos(X,R1),!,
elementos(Cola,R2),
concatenar(R1,R2,R).*

- elementos([X | Cola],R):- elementos(Cola,R1),
concatenar([X],R1,R).*

- ?- elementos([a,b,c,[d,e],f],R).*

- R = [a, b, c, d, e, f].*

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplos:
 1. Máximo
 2. Máximo común divisor
 3. Crear lista de enteros
 4. Elementos
 - 5. Extremos**
 6. Suprimir
 7. Préstamo de libros

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**

- Ejemplos 5.- Extremos

extremos([],[]):-!.

*extremos(L,[P,U]):- primero(L,P),
ultimo(L,U).*

primero([],[]):-!.

primero([X|_],X).

ultimo([],[]):-!.

ultimo([X],X):-!.

ultimo([_ | Cola],R):- ultimo(Cola,R).

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplos 5.- Extremos
?- *extremos([a,b,c,d,e],R).*
R = [a, e].

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplos:
 1. Máximo
 2. Máximo común divisor
 3. Crear lista de enteros
 4. Elementos
 5. Extremos
 6. **Suprimir**
 7. Préstamo de libros

2. Predicados de control de ejecución

- **El corte: !**

- **Confirma el uso de una regla**

- **Ejemplo 6.- Suprimir**

- *Elimina un elemento X de una lista simple*
suprimir(_, [], []).

suprimir(X, [X | Cola], R):- !, suprimir(X, Cola, R).

suprimir(X, [Y | Cola], [Y | R]):- suprimir(X, Cola, R).

?- suprimir(a, [a, b, a, c, d, a], R).

R = [b, c, d]

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplos:
 1. Máximo
 2. Máximo común divisor
 3. Crear lista de enteros
 4. Elementos
 5. Extremos
 6. Suprimir
 7. Préstamo de libros

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplo 7.- Préstamo de libros

/ Tipos de servicios */*

servicio_basico(consulta).

servicio_basico(referencia).

servicio_adicional(prestamo).

servicio_adicional(prestamo_interbibliotecario).

servicio_general(X):- servicio_basico(X).

servicio_general(X):- servicio_adicional(X).

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplo 7.- Préstamo de libros

/ Servicios de personas */*

```
servicio(Persona, Servicio):-  
    lector(Persona),  
    prestamo(Persona, _),  
    !,  
    servicio_basico(Servicio).
```

```
servicio(Persona, Servicio):-  
    lector(Persona),  
    servicio_general(Servicio).
```

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - Ejemplo 7.- Préstamo de libros

/ Lectores */*

lector('Juan Campos Aguilera').

lector('Ana Silva Arroyo').

lector('Pedro Luque Salas').

/ Préstamos */*

prestamo('Juan Campos Aguilera','Nazarín').

prestamo('Juan Campos Aguilera','Misericordia').

prestamo('Ana Silva Arroyo','La cruz del Sur').

2. Predicados de control de ejecución

- **El corte: !**
 - **Confirma el uso de una regla**
 - **Ejemplo 7.- Préstamo de libros**

/ La pregunta */*

?- servicio(P,S).

*P = 'Juan Campos Aguilera',
S = consulta ;*

*P = 'Juan Campos Aguilera',
S = referencia.*

2. Predicados de control de ejecución

- **El corte: !**
 - **Aplicaciones**
 - Evita la búsqueda de soluciones alternativas inexistentes.
 - Confirma el uso de una regla
 - **Provoca un fallo inmediato: corte y *fail***

2. Predicados de control de ejecución

- **El corte: !**
 - **Provoca un fallo inmediato: corte y *fail***
 - Ejemplos:
 1. Vegetarianos
 2. Personas y animales

2. Predicados de control de ejecución

- **El corte: !**
 - **Provoca un fallo inmediato: corte y *fail***
 - Ejemplos:
 1. Vegetarianos
 2. Personas y animales

2. Predicados de control de ejecución

- El corte: !
 - Provoca un fallo inmediato: **corte** y **fail**
 - Ejemplo 1.- Vegetarianos

come(Persona,X):-

vegetariano(Persona),

not(verdura(X)),

!,

fail.

come(_,X):- comida(X).

comida(X):- carne(X) ; verdura(X) ; pescado(X).

2. Predicados de control de ejecución

- El corte: !

- Provoca un fallo inmediato: **corte** y **fail**

- Ejemplo 1.- Vegetarianos

verdura(espinacas).

verdura(acelgas).

carne(cerdo).

carne(ternera).

pescado(bacalao).

pescado(merluza).

vegetariano(anselmo).

vegetariano(alicia).

? come(anselmo, cerdo).

No

2. Predicados de control de ejecución

- **El corte: !**
 - **Provoca un fallo inmediato: **corte** y *fail***
 - Ejemplos:
 1. Vegetarianos
 2. **Personas y animales**

2. Predicados de control de ejecución

- El corte: !

- Provoca un fallo inmediato: **corte** y **fail**

- Ejemplo 2.- Personas y animales

le_gusta(maria, X):- serpiente(X),

!,

fail.

le_gusta(maria, X):- animal(X).

ave(loro).

ave(buitre).

serpiente(boa).

serpiente(anaconda).

animal(X):- serpiente(X); ave(X).

2. Predicados de control de ejecución

- *true*
- *fail*
- *repeat*
- El corte: !
- *call*
- El condicional: ->

2. Predicados de control de ejecución

- *call*

- Sintaxis

call(predicado)

- Significado

- Permite ejecutar otro predicado

- Ejemplo

?- *call(write(('Hola'))).*

Hola

?- *call((write('hola'), write(' '), write('amigo'))).*

hola amigo

true.

2. Predicados de control de ejecución

- *call*
 - Observación
 - Junto con el corte *!* y *fail*, permite la definición de “*not*”.

not(X) :- call(X), !, fail.

not(X).

2. Predicados de control de ejecución

- *call*
 - **Ejemplos**
 - Filtrar una lista por cualquier predicado.
 - Aplicar un predicado a los elemento de una lista.

2. Predicados de control de ejecución

- *call*

- **Ejemplos**

- Filtrar una lista por cualquier predicado: versión 1

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

`filtrar(Predicado, [Cabeza | Cola], [Cabeza | Cola1]):-`
`call(Predicado, Cabeza),`
`filtrar(Predicado, Cola, Cola1).`

`filtrar(Predicado, [Cabeza | Cola], Cola1):-`
`not(call(Predicado, Cabeza)),`
`filtrar(Predicado, Cola, Cola1).`

2. Predicados de control de ejecución

- *call*

- **Ejemplos**

- Filtrar una lista por cualquier predicado: versión 1

?- filtrar(impar,[1,2,3,4,5],R).

R = [1, 3, 5] .

2. Predicados de control de ejecución

- *call*
 - Ejemplos
 - Filtrar una lista por cualquier predicado: versión 2

filtrarV2(**Predicado**, Lista, Resultado):-

findall(X, (member(X, Lista), **call**(**Predicado**, X)), Resultado).

?- filtrarV2(impar,[1,2,3,4,5],R).

R = [1, 3, 5].

2. Predicados de control de ejecución

- *call*

- **Ejemplos**

- Aplicar un predicado a los elemento de una lista:
versión 1

aplicar(_ , [] , []).

aplicar(**Predicado**, [Cabeza|Cola], [Resultado|Cola1]):-

call(**Predicado**, Cabeza, Resultado),

aplicar(**Predicado**, Cola, Cola1).

?- aplicar(sqrt,[1,2,3,4,5],R).

R = [1.0, 1.4142135623730951, 1.7320508075688772, 2.0,
2.23606797749979] .

2. Predicados de control de ejecución

- *call*

- **Ejemplos**

- Aplicar un predicado a los elemento de una lista:
versión 2

aplicarV2(**Predicado**, ListaOriginal, ListaResultante):-

findall(Y,

(member(X, ListaOriginal), **call**(**Predicado**, X, Y)),

ListaResultante).

?- aplicarV2(sqrt,[1,2,3,4,5],R).

R = [1.0, 1.4142135623730951, 1.7320508075688772, 2.0,
2.23606797749979].

2. Predicados de control de ejecución

- *true*
- *fail*
- *repeat*
- *El corte: !*
- *call*
- **El condicional: ->**

2. Predicados de control de ejecución

- El condicional: \rightarrow

- Sintaxis

predicado \rightarrow *consecuente* ; *alternativa*

- Descripción

- Si el *predicado* es cierto, se ejecuta el *consecuente*; en caso contrario, la *alternativa*.
- Implementa el “corte suave” (*soft cut*).

$$P \rightarrow Q :- P, !, Q.$$

- Es un operador infijo y asociativo por la derecha.

2. Predicados de control de ejecución

- El condicional: \rightarrow

- Ejemplos

nota(X,Y) :-

X < 5 \rightarrow Y = suspenso ;

X < 7 \rightarrow Y = aprobado ;

X < 9 \rightarrow Y = notable ;

true \rightarrow Y = sobresaliente.

?- nota(7,Y).

Y = notable.

2. Predicados de control de ejecución

- El condicional: ->

- Ejemplos

- Equivalencia (1/2)

nota(X,Y) :- X < 5, Y = suspenso.

nota(X,Y) :- X >= 5, X < 7, Y = aprobado.

nota(X,Y) :- X >= 7, X < 9, Y = notable.

nota(X,Y) :- X >= 9, Y = sobresaliente.

?- nota(7,Y).

Y = notable.

2. Predicados de control de ejecución

- **El condicional: ->**

- **Ejemplos**

- **Equivalencia (2/2)**

nota(X, suspenso) :- X < 5.

nota(X, aprobado) :- X >= 5, X < 7.

nota(X, notable) :- X >= 7, X < 9.

nota(X, sobresaliente) :- X >= 9.

?- nota(7, Y).

Y = notable.

2. Predicados de control de ejecución

- El condicional: ->

- Ejemplos

- Menú (1/2)

menu :-

```
write("Ejemplo del menú"),  
repeat,  
write("Elija una opción"),nl,  
write("1 -> Primera "),nl,  
write("2 -> Segunda "),nl,  
write("3 -> Tercera "),nl, nl,  
write("0 -> Fin "),nl,  
read(Op),
```

2. Predicados de control de ejecución

- El condicional: ->

- Ejemplos

- Menú (2/2)

```
(  
    Op ::= 0 -> write("Fin del ejemplo del menú");  
    Op ::= 1 -> write("Primera opción "),nl, fail;  
    Op ::= 2 -> write("Segunda opción "),nl, fail;  
    Op ::= 3 -> write("Tercera opción "),nl, fail;  
    true -> write("Opción incorrecta -->"),  
            write(Op), nl, nl, fail  
)  
nl.
```



UNIVERSIDAD DE CÓRDOBA

ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA

DEPARTAMENTO DE
INFORMÁTICA Y ANÁLISIS NUMÉRICO



PROGRAMACIÓN DECLARATIVA

INGENIERÍA INFORMÁTICA

CUARTO CURSO

PRIMER CUATRIMESTRE

Tema 11.- Reevaluación y el “corte”

