

Lenguajes, Tecnologías y Paradigmas de la programación (LTP)

Práctica 7: Introducción a PROLOG



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Sergio Pérez
serperu@dsic.upv.es

Introducción a PROLOG

OBJETIVOS DE LA PRÁCTICA

SWI-Prolog:

- Cargar y ejecutar programas
- Consultas a hechos
- Consultas a reglas

Conceptos básicos de Prolog:

- Unificación de términos
- Búsqueda con retroceso (Backtracking)

Aritmética simple en Prolog

Comandos básicos

Abrir SWI-Prolog:

```
bash$ swipl
```

```
...
```

```
?-
```

```
?- halt.    % Salir de SWI-Prolog
```

```
bash$
```

Comandos básicos

Abrir SWI-Prolog:

```
bash$ swipl                                     ?- halt.    % Salir de SWI-Prolog
...                                              bash$
?- 
```

Cargar un fichero “.pl” :

```
?- consult(FICHERO_SIN_EXTENSION).
?- compile(FICHERO_SIN_EXTENSION).
?- [FICHERO_SIN_EXTENSION].
```

Comandos básicos

Abrir SWI-Prolog:

```
bash$ swipl                                ?- halt.    % Salir de SWI-Prolog
...                                         bash$
?-
```

Cargar un fichero “.pl” :

```
?- consult(FICHERO_SIN_EXTENSION).
?- compile(FICHERO_SIN_EXTENSION).      → true.
?- [FICHERO_SIN_EXTENSION].
```

Comandos básicos

```
?- [cars].  
true.
```

Comandos básicos

```
?- [cars].  
true.
```

Mostrar los datos del fichero cargado:

Comandos básicos

```
?- [cars].  
true.
```

Mostrar los datos del fichero cargado:

```
?- listing.
```


Comandos básicos

```
?- [cars].  
true.
```

Mostrar los datos del fichero cargado:

```
?- listing.  
model(ibiza, seat).  
model(cordoba, seat).  
model(altea, seat).
```

...

Hechos, reglas y tipos de predicados

- Hechos:

```
model(ibiza, seat).
```

Hechos, reglas y tipos de predicados

- Hechos:

`model(ibiza, seat).`

- Reglas:

`brand(A,B) :- model(B,A).`

Hechos, reglas y tipos de predicados

- Hechos:

`model(ibiza, seat).`

- Reglas:

`brand(A,B) :- model(B,A).`

- Predicados extensionales: Explícitos en el fichero.

`model(ibiza, seat).`

Hechos, reglas y tipos de predicados

- Hechos:

`model(ibiza, seat).`

- Reglas:

`brand(A,B) :- model(B,A).`

- Predicados **extensionales**: Explícitos en el fichero.

`model(ibiza, seat).`

- Predicados **intensionales**: Obtenidos ejecutando las reglas.

`brand(A,B) :- model(B,A).`

`model(ibiza, seat).`

Hechos, reglas y tipos de predicados

- Hechos:

`model(ibiza, seat).`

- Reglas:

`brand(A,B) :- model(B,A).`

- Predicados **extensionales**: Explícitos en el fichero.

`model(ibiza, seat).`

- Predicados **intensionales**: Obtenidos ejecutando las reglas.

`brand(A,B) :- model(B,A).`

`model(ibiza, seat).`

→ `brand(seat, ibiza).`

Consultas

```
model(ibiza,seat).
```

```
model(cordoba,seat).
```

```
brand(A,B) :- model(B,A).
```

Consultas

```
model(ibiza,seat).  
model(cordoba,seat).  
brand(A,B) :- model(B,A).
```

Es equivalente a:

Consultas

```
model(ibiza,seat).  
model(cordoba,seat).  
brand(A,B) :- model(B,A).
```

Es equivalente a:

```
model(ibiza,seat).  
model(cordoba,seat).  
brand(seat,ibiza).  
brand(seat,cordoba).
```

Consultas

```
model(ibiza,seat).    brand(A,B) :- model(B,A).  
model(cordoba,seat).
```

Consultas

```
model(ibiza,seat).    brand(A,B) :- model(B,A).  
model(cordoba,seat).
```

Consultas a hechos:

```
% Todas las marcas  
% que tiene un modelo  
% llamado ibiza
```

Consultas

```
model(ibiza,seat).    brand(A,B) :- model(B,A).  
model(cordoba,seat).
```

Consultas a hechos:

```
% Todas las marcas  
% que tiene un modelo  
% llamado ibiza
```

```
?- model(ibiza, X).  
X = seat.
```

Consultas

```
model(ibiza,seat).    brand(A,B) :- model(B,A).  
model(cordoba,seat).
```

Consultas a hechos:

```
% Todas las marcas  
% que tiene un modelo  
% llamado ibiza
```

```
?- model(ibiza, X).  
X = seat.
```

Consultas a reglas:

```
% Todos los modelos de  
% la marca seat
```

Consultas

```
model(ibiza,seat).    brand(A,B) :- model(B,A).  
model(cordoba,seat).
```

Consultas a hechos:

```
% Todas las marcas  
% que tiene un modelo  
% llamado ibiza
```

```
?- model(ibiza, X).  
X = seat.
```

Consultas a reglas:

```
% Todos los modelos de  
% la marca seat
```

```
?- brand(seat,X).
```

Consultas

```
model(ibiza,seat).    brand(A,B) :- model(B,A).  
model(cordoba,seat).
```

Consultas a hechos:

```
% Todas las marcas  
% que tiene un modelo  
% llamado ibiza
```

```
?- model(ibiza, X).  
X = seat.
```

Consultas a reglas:

```
% Todos los modelos de  
% la marca seat
```

```
?- brand(seat,X).  
X = ibiza.  
X = cordoba.
```

Reglas y cláusulas

model(ibiza,seat).
model(cordoba,seat).

segment(ibiza,c).
segment(panda,c).

Reglas y cláusulas

`model(ibiza,seat).`
`model(cordoba,seat).`

`segment(ibiza,c).`
`segment(panda,c).`

- Regla:

Reglas y cláusulas

`model(ibiza,seat).`

`model(cordoba,seat).`

`segment(ibiza,c).`

`segment(panda,c).`

- Regla:

`isRelated(A,B) :- model(A,C), model(B,C), A \== B.`

Reglas y cláusulas

`model(ibiza,seat).`

`model(cordoba,seat).`

`segment(ibiza,c).`

`segment(panda,c).`

- Regla:

`isRelated(A,B) :- model(A,C), model(B,C), A \== B.`

`isRelated(A,B) :- segment(A,S), segment(B,S), A \== B.`

Reglas y cláusulas

`model(ibiza,seat).`

`model(cordoba,seat).`

`segment(ibiza,c).`

`segment(panda,c).`

- Regla:

`isRelated(A,B) :- model(A,C), model(B,C), A \== B.`

`isRelated(A,B) :- segment(A,S), segment(B,S), A \== B.`

`?- isRelated(ibiza,X)`

Reglas y cláusulas

`model(ibiza,seat).`

`model(cordoba,seat).`

`segment(ibiza,c).`

`segment(panda,c).`

- Regla:

`isRelated(A,B) :- model(A,C), model(B,C), A \== B.`

`isRelated(A,B) :- segment(A,S), segment(B,S), A \== B.`

`?- isRelated(ibiza,X)`

`X = cordoba.` → Cláusula 1 (model)

Reglas y cláusulas

`model(ibiza,seat).`

`model(cordoba,seat).`

`segment(ibiza,c).`

`segment(panda,c).`

- Regla:

`isRelated(A,B) :- model(A,C), model(B,C), A \== B.`

`isRelated(A,B) :- segment(A,S), segment(B,S), A \== B.`

`?- isRelated(ibiza,X)`


`X = cordoba.` → Cláusula 1 (model)

`X = panda.` → Cláusula 2 (segment)



Unificación de términos

Término 1	¿Unificación?	Término 2
date(10,nov,2030)		date(10,nov,2030)
date(10,nov,2030)		date(X,nov,2030)
date(10,nov,2030)		date(X,nov,X)
date(10,nov,2030)		time(13,05)
date(10,nov,2030)		X


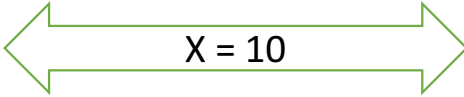
Unificación de términos

Término 1	¿Unificación?	Término 2
date(10,nov,2030)		date(10,nov,2030)
date(10,nov,2030)		date(X,nov,2030)
date(10,nov,2030)		date(X,nov,X)
date(10,nov,2030)		time(13,05)
date(10,nov,2030)		X


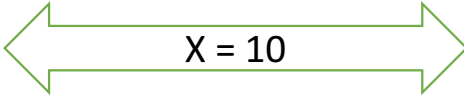
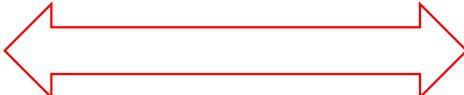
Unificación de términos

Término 1	¿Unificación?	Término 2
date(10,nov,2030)		date(10,nov,2030)
date(10,nov,2030)		date(X,nov,2030)
date(10,nov,2030)		date(X,nov,X)
date(10,nov,2030)		time(13,05)
date(10,nov,2030)		X


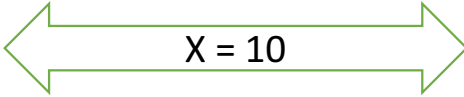
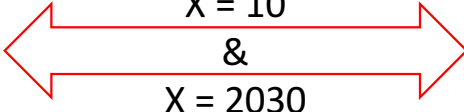
Unificación de términos

Término 1	¿Unificación?	Término 2
date(10,nov,2030)		date(10,nov,2030)
date(10,nov,2030)		date(X,nov,2030)
date(10,nov,2030)		date(X,nov,X)
date(10,nov,2030)		time(13,05)
date(10,nov,2030)		X


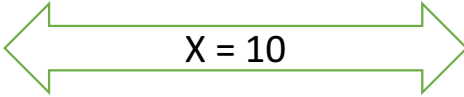
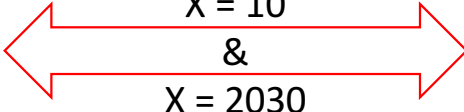
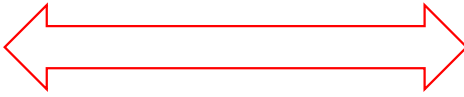
Unificación de términos

Término 1	¿Unificación?	Término 2
date(10,nov,2030)		date(10,nov,2030)
date(10,nov,2030)		date(X,nov,2030)
date(10,nov,2030)		date(X,nov,X)
date(10,nov,2030)		time(13,05)
date(10,nov,2030)		X


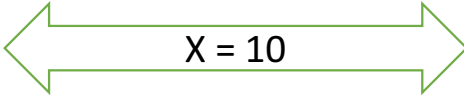
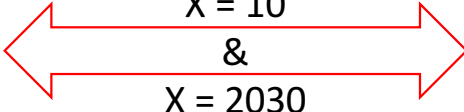
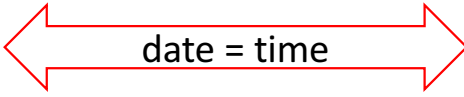
Unificación de términos

Término 1	¿Unificación?	Término 2
date(10,nov,2030)		date(10,nov,2030)
date(10,nov,2030)		date(X,nov,2030)
date(10,nov,2030)		date(X,nov,X)
date(10,nov,2030)		time(13,05)
date(10,nov,2030)		X


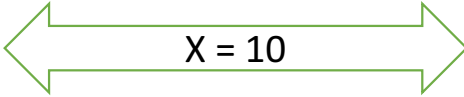
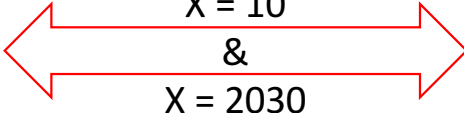
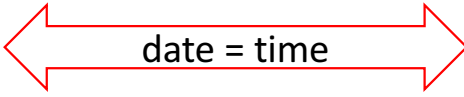
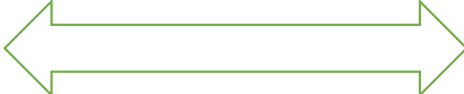
Unificación de términos

Término 1	¿Unificación?	Término 2
date(10,nov,2030)		date(10,nov,2030)
date(10,nov,2030)		date(X,nov,2030)
date(10,nov,2030)		date(X,nov,X)
date(10,nov,2030)		time(13,05)
date(10,nov,2030)		X


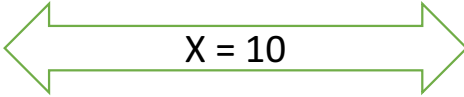
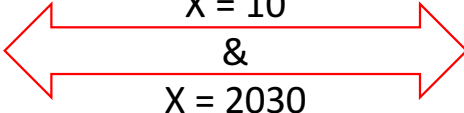
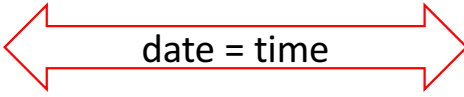
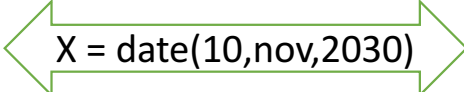
Unificación de términos

Término 1	¿Unificación?	Término 2
date(10,nov,2030)		date(10,nov,2030)
date(10,nov,2030)		date(X,nov,2030)
date(10,nov,2030)		date(X,nov,X)
date(10,nov,2030)		time(13,05)
date(10,nov,2030)		X

Unificación de términos

Término 1	¿Unificación?	Término 2
date(10,nov,2030)		date(10,nov,2030)
date(10,nov,2030)		date(X,nov,2030)
date(10,nov,2030)		date(X,nov,X)
date(10,nov,2030)		time(13,05)
date(10,nov,2030)		X

Unificación de términos

Término 1	¿Unificación?	Término 2
date(10,nov,2030)		date(10,nov,2030)
date(10,nov,2030)		date(X,nov,2030)
date(10,nov,2030)		date(X,nov,X)
date(10,nov,2030)		time(13,05)
date(10,nov,2030)		X

Activar/desactivar el modo traza/depuración

```
?- trace.
```

```
[trace] ?-
```

Activar/desactivar el modo traza/depuración

```
?- trace.
```

```
[trace] ?-
```

```
?- debug.
```

```
[debug] ?-
```

Activar/desactivar el modo traza/depuración

```
?- trace.
```

```
[trace] ?-
```

```
[trace] ?- notrace.
```

```
?- debug.
```

```
[debug] ?-
```

Activar/desactivar el modo traza/depuración

```
?- trace.
```

```
[trace] ?-
```

```
[trace] ?- notrace.
```

```
[debug] ?-
```

```
?- debug.
```

```
[debug] ?-
```

Activar/desactivar el modo traza/depuración

```
?- trace.
```

```
[trace] ?-
```

```
[trace] ?- notrace.
```

```
[debug] ?- nodebug.
```

```
?-
```

```
?- debug.
```

```
[debug] ?-
```

Backtracking (algoritmo)

- 1) Buscar hecho que unifique con la consulta
- 2) Buscar cabeza de regla que unifique con la consulta
- 3) Si existe, ejecutar 1) por cada cláusula
- 4) Fail

Backtracking (ejemplo)

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).          model(cordoba,seat).
```

Backtracking (ejemplo)

?- isRelated(ibiza, X).

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).          model(cordoba,seat).
```


Backtracking (ejemplo)

```
?- isRelated(ibiza, X).
```

```
Call: (8) isRelated(ibiza, _10636) ? creep
```

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).           model(cordoba,seat).
```

Backtracking (ejemplo)

?- isRelated(ibiza, X).

Call: (8) isRelated(ibiza, _10636) ? creep

Call: (9) model(ibiza, _10872) ? creep

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).           model(cordoba,seat).
```

Backtracking (ejemplo)

?- isRelated(ibiza, X).

Call: (8) isRelated(ibiza, _10636) ? creep

Call: (9) model(ibiza, _10872) ? creep

Exit: (9) model(ibiza, seat) ? creep

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).           model(cordoba,seat).
```

Backtracking (ejemplo)

?- isRelated(ibiza, X).

Call: (8) isRelated(ibiza, _10636) ? creep

Call: (9) model(ibiza, _10872) ? creep

Exit: (9) model(ibiza, seat) ? creep

Redo: (9) model(_10636, seat) ? creep

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).           model(cordoba,seat).
```

Backtracking (ejemplo)

?- isRelated(ibiza, X).

Call: (8) isRelated(ibiza, _10636) ? creep

Call: (9) model(ibiza, _10872) ? creep

Exit: (9) model(ibiza, seat) ? creep

Redo: (9) model(_10636, seat) ? creep

Exit: (9) model(ibiza, seat) ? creep

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).           model(cordoba,seat).
```

Backtracking (ejemplo)

```
?- isRelated(ibiza, X).
```

Call: (8) isRelated(ibiza, _10636) ? creep

Call: (9) model(**ibiza**, **_10872**) ? creep

```
Exit: (9) model(ibiza, seat) ? creep
```

Redo: (9) `model(_10636, seat) ? creep`

```
Exit: (9) model(ibiza, seat) ? creep
```

Call: (9) **ibiza**\==**ibiza**? creep

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).          model(cordoba,seat).
```

Backtracking (ejemplo)

```
?- isRelated(ibiza, X).
```

Call: (8) `isRelated(ibiza, _10636) ? creep`

Call: (9) model(**ibiza**, **_10872**) ? creep

```
Exit: (9) model(ibiza, seat) ? creep
```

Redo: (9) `model(_10636, seat) ? creep`

```
Exit: (9) model(ibiza, seat) ? creep
```

Call: (9) **ibiza**\==**ibiza**? creep

Fail: (9) `ibiza\==ibiza?` creep

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).          model(cordoba,seat).
```

Backtracking (ejemplo)

```
?- isRelated(ibiza, X).
```

Call: (8) isRelated(ibiza, 10636) ? creep

Call: (9) model(**ibiza**, **_10872**) ? creep

```
Exit: (9) model(ibiza, seat) ? creep
```

Redo: (9) model(10636, seat) ? creep

```
Exit: (9) model(ibiza, seat) ? creep
```

Call: (9) **ibiza**\==**ibiza**? creep

Fail: (9) `ibiza\==ibiza?` creep

Redo: (9) model(10636, seat) ? creep

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).          model(cordoba,seat).
```


Backtracking (ejemplo)

Redo: (9) model(_10636, seat) ? creep

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).  
model(cordoba,seat).
```

Backtracking (ejemplo)

Redo: (9) model(_10636, seat) ? creep

Exit: (9) model(cordoba, seat) ? creep

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).  
model(cordoba,seat).
```

Backtracking (ejemplo)

Redo: (9) model(_10636, seat) ? creep

Exit: (9) model(cordoba, seat) ? creep

Call: (9) ibiza\==cordoba ? creep

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).  
model(cordoba,seat).
```

Backtracking (ejemplo)

Redo: (9) model(_10636, seat) ? creep

Exit: (9) model(cordoba, seat) ? creep

Call: (9) ibiza \== cordoba ? creep

Exit: (9) ibiza \== cordoba ? creep

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).  
model(cordoba,seat).
```

Backtracking (ejemplo)

Redo: (9) `model(_10636, seat) ? creep`

Exit: (9) `model(cordoba, seat) ? creep`

Call: (9) `ibiza\==cordoba ? creep`

Exit: (9) `ibiza\==cordoba ? creep`

Exit: (8) `isRelated(ibiza, cordoba) ? creep`

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).  
model(cordoba,seat).
```

Backtracking (ejemplo)

Redo: (9) `model(_10636, seat)` ? creep

Exit: (9) `model(cordoba, seat)` ? creep

Call: (9) `ibiza\==cordoba` ? creep

Exit: (9) `ibiza\==cordoba` ? creep

Exit: (8) `isRelated(ibiza, cordoba)` ? creep

X = cordoba.

```
isRelated(A,B) :- model(A,C), model(B,C), A \== B.  
model(ibiza,seat).  
model(cordoba,seat).
```

Aritmética simple en Prolog

- Diferencia entre Unificación (=) y Evaluación (is)

Aritmética simple en Prolog

- Diferencia entre Unificación (=) y Evaluación (is)

- $X = 1 + 2$ unifica:

- $X = +(1,2)$

% (Notación prefija)

Aritmética simple en Prolog

- Diferencia entre Unificación (=) y Evaluación (is)

- $X = 1 + 2$ unifica:

- $X = +(1,2)$

- % (Notación prefija)

- $X \text{ is } 1 + 2$ evalúa forzosamente $1 + 2$ y lo unifica con X

- $X = 3$

Aritmética simple en Prolog

- Diferencia entre Unificación (=) y Evaluación (is)
 - $X = 1 + 2$ unifica:
 $X = +(1,2)$
% (Notación prefija)
 - $X \text{ is } 1 + 2$ evalúa forzosamente $1 + 2$ y lo unifica con X
 $X = 3$
- $X \text{ is } X + 1$

Aritmética simple en Prolog

- Diferencia entre Unificación (=) y Evaluación (is)
 - $X = 1 + 2$ unifica:
 $X = +(1,2)$
% (Notación prefija)
 - $X \text{ is } 1 + 2$ evalúa forzosamente $1 + 2$ y lo unifica con X
 $X = 3$
- $X \text{ is } X + 1$ da error de ejecución

Aritmética simple en Prolog

- Diferencia entre Unificación (=) y Evaluación (is)
 - $X = 1 + 2$ unifica:
 $X = +(1,2)$
% (Notación prefija)
 - $X \text{ is } 1 + 2$ evalúa forzosamente $1 + 2$ y lo unifica con X
 $X = 3$
- $X \text{ is } X + 1$ da error de ejecución. X no puede tener 2 valores distintos en la misma regla

Aritmética simple en Prolog

- Diferencia entre Unificación (=) y Evaluación (is)
 - $X = 1 + 2$ unifica:
 $X = +(1,2)$
% (Notación prefija)
 - $X \text{ is } 1 + 2$ evalúa forzosamente $1 + 2$ y lo unifica con X
 $X = 3$
- $X \text{ is } X + 1$ da error de ejecución. X no puede tener 2 valores distintos en la misma regla. **Solución:** $X2 \text{ is } X + 1$

Aritmética simple en Prolog

- Diferencia entre Unificación (=) y Evaluación (is)
 - $X = 1 + 2$ unifica:
 $X = +(1,2)$
% (Notación prefija)
 - $X \text{ is } 1 + 2$ evalúa forzosamente $1 + 2$ y lo unifica con X
 $X = 3$
- $X \text{ is } X + 1$ da error de ejecución. X no puede tener 2 valores distintos en la misma regla. **Solución:** $X2 \text{ is } X + 1$
- $3 \text{ is } Y + 2$ falla si Y no tiene un valor previamente:

Aritmética simple en Prolog

- Diferencia entre Unificación (=) y Evaluación (is)
 - $X = 1 + 2$ unifica:
 $X = +(1,2)$
% (Notación prefija)
 - $X \text{ is } 1 + 2$ evalúa forzosamente $1 + 2$ y lo unifica con X
 $X = 3$
 - $X \text{ is } X + 1$ da error de ejecución. X no puede tener 2 valores distintos en la misma regla. **Solución:** $X2 \text{ is } X + 1$
 - $3 \text{ is } Y + 2$ falla si Y no tiene un valor previamente:
(1) `rule(Y,X) :- X is Y + 1`


Aritmética simple en Prolog

- Diferencia entre Unificación (=) y Evaluación (is)
 - $X = 1 + 2$ unifica:
 $X = +(1,2)$
% (Notación prefija)
 - $X \text{ is } 1 + 2$ evalúa forzosamente $1 + 2$ y lo unifica con X
 $X = 3$
- $X \text{ is } X + 1$ da error de ejecución. X no puede tener 2 valores distintos en la misma regla. **Solución:** $X2 \text{ is } X + 1$
- $3 \text{ is } Y + 2$ falla si Y no tiene un valor previamente:
 - (1) `rule(Y,X) :- X is Y + 1`
 - (2) `rule(A,X) :- X is Y + 1`

Aritmética simple en Prolog

- Diferencia entre Unificación (=) y Evaluación (is)
 - $X = 1 + 2$ unifica:
 $X = +(1,2)$
% (Notación prefija)
 - $X \text{ is } 1 + 2$ evalúa forzosamente $1 + 2$ y lo unifica con X
 $X = 3$
 - $X \text{ is } X + 1$ da error de ejecución. X no puede tener 2 valores distintos en la misma regla. **Solución:** $X2 \text{ is } X + 1$
 - $3 \text{ is } Y + 2$ falla si Y no tiene un valor previamente:
(1) $\text{rule}(Y,X) \text{ :- } X \text{ is } Y + 1$
(2) $\text{rule}(A,X) \text{ :- } X \text{ is } Y + 1$ ¿ $\text{rule}(1,X)$?

Aritmética simple en Prolog

- Diferencia entre Unificación (=) y Evaluación (is)
 - $X = 1 + 2$ unifica:
 $X = +(1,2)$
% (Notación prefija)
 - $X \text{ is } 1 + 2$ evalúa forzosamente $1 + 2$ y lo unifica con X
 $X = 3$
 - $X \text{ is } X + 1$ da error de ejecución. X no puede tener 2 valores distintos en la misma regla. **Solución:** $X2 \text{ is } X + 1$
 - $3 \text{ is } Y + 2$ falla si Y no tiene un valor previamente:
(1) $\text{rule}(Y,X) \text{ :- } X \text{ is } Y + 1$ 
(2) $\text{rule}(A,X) \text{ :- } X \text{ is } Y + 1$ ¿ $\text{rule}(1,X)$?

Aritmética simple en Prolog

- Diferencia entre Unificación (=) y Evaluación (is)
 - $X = 1 + 2$ unifica:
 $X = +(1,2)$
% (Notación prefija)
 - $X \text{ is } 1 + 2$ evalúa forzosamente $1 + 2$ y lo unifica con X
 $X = 3$
 - $X \text{ is } X + 1$ da error de ejecución. X no puede tener 2 valores distintos en la misma regla. **Solución:** $X2 \text{ is } X + 1$
 - $3 \text{ is } Y + 2$ falla si Y no tiene un valor previamente:
(1) $\text{rule}(Y,X) \text{ :- } X \text{ is } Y + 1$ 🧐
(2) $\text{rule}(A,X) \text{ :- } X \text{ is } Y + 1$ 🙅 ¿rule(1,X)?