PRÁCTICAS DE MATEMÁTICAS 1

2018-2019

- Repaso sintaxis Prolog. Programas.
- Acciones de Plman.
- Resolver mapas: plman/maps/ejemplos

- Vídeos d Prácticas M1. curso 2017-18.
- Prof. Francisco Gallego. https://bit.ly/prácticasM1

>>>>> Consultar esta transpa para las prácticas 1 y 2 de Lógica vs Prolog.
>>>>> No ejecutar los pasos para programas de Plman

PASOS para ESCRIBIR/EJECUTAR PROGRAMAS Prolog con SWI-PROLOG

>> **Abrir** terminal **Linux**

>> **Teclear** comando para abrir intérprete Swi-Prolog

$ swipl

>> **Editar** fichero, editores: emacs, gedit…

? emacs('fichero.pl').

>> **Escribir programa** Prolog > crear base de conocimiento.

>> **Compilar /cargar** en memoria el fichero : fichero.pl.

- si está editado hacer desde el **menú edición**: **Compile /Compile buffer**
- si no se ha editado: **? consult('fichero.pl').**

>> **Ejecutar** programa: **? pregunta.**

LÓGICA VS PROLOG

PROGRAMA PROLOG

HECHOS: predicado(arg1, arg2,...., argN). **argi: constante / número**

También argi puede ser: predicado(arg) que es una estructura en Prolog llamada "fórmula atómica" y es equivalente a un átomo, pero con estructura. Por lo tanto, se considera equivalente a un elemento atómico del dominio en Lógica de Predicados

REGLAS: condicional

cabeza :- cuerpo *(cabeza cierta SI cuerpo cierto)*

Cabeza: máximo un predicado con N ≥ 0 argumentos.

Cuerpo: conjunción de M predicados, M ≥ 1.

En los argumentos pueden aparecer variables.

COMENTARIOS

```
% Comentarios
/* Comentario
de 2 líneas */
```

SINTAXIS en LÓGICA

PREDICADO: **predicado(arg1, arg2,...., argN).**

- Cualquier identificador de nombre, por lo general en mayúscula Ej. G(x)

argi: constante (átomo): identifica de forma única a un objeto del dominio.

- Cualquier identificador de nombre propio o bien: a, b, c….

argi: variable: identifica de forma genérica a un objeto del dominio:

- Cualquier identificador de variable matemática: x, y, z.

SINTAXIS en PROLOG

PREDICADO: **predicado(arg1, arg2,...., argN).**

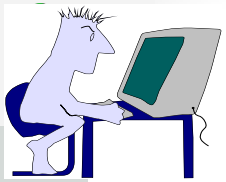
- Empieza por **letra minúscula**.
- No hay espacio entre el nombre y el paréntesis.
- Los argumentos (términos), separados por comas.

argi: constante (átomo): identifica de forma única a un objeto del dominio.

- Empieza por letra **minúscula.**
- Puede contener letras, números o subrayado.
- **No** puede contener espacios.
- Todo lo que vaya entre comillas simples ' ' .

argi: variable: identifica de forma genérica a un objeto del dominio:

- Empieza por **letra mayúscula** o **subrayado.**
- Puede contener letras, números o subrayado.
- No puede contener espacios.



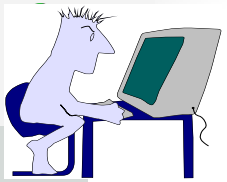
Práctica 1-Lógica vs Prolog

Escribir en fichero **'raz1.pl'.**
Formalizar en L. predicados y en Prolog
Ejecutar las preguntas

Escribir un programa en PROLOG que conteste a la pregunta planteada en el razonamiento-1:

P1: Todos los alum tienen sentido del humor
P2: Juan es un alum

Responder Q**: ¿ Juan tiene sentido del humor ?**



Práctica 1-Lógica vs Prolog (cont)

PASOS:

P1: Todos los alum tienen sentido del humor

P2: Juan es un alum

Responder Q: **¿Juan tiene sentido del humor?**

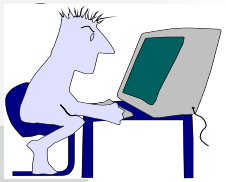
Predicados: ⇨

sentidoH(X): X es un sujeto que tiene sentido humor

alum(X): X es un sujeto que tiene la propiedad de ser alum

Formalización

Lógica predicados	**Prolog**
$\forall x\,[alum(x) \rightarrow sentidoH(x)]$	sentidoH(X) :- alum(X).



Práctica 1-Lógica vs Prolog (cont)

PASOS:

P1: Todos los alum tienen sentido del humor

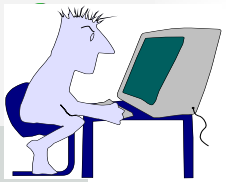
P2: Juan es un alum

Responder Q : **¿Juan tiene sentido del humor?**

P2: es proposición atómica,
es un hecho en Prolog

Formalización

Lógica predicados	**Prolog**
alum(juan)	**alum(juan).**



Práctica 1-Lógica vs Prolog (cont)

PASOS:

P1: Todos los alum tienen sentido del humor
P2: Juan es un alum

Responder Q : ¿**Juan tiene sentido del humor**?

Q: es proposición atómica,
es un hecho en Prolog

Formalización

Lógica predicados	**Prolog**
sentidoH(juan)	**sentidoH(juan).**

Práctica 2-Lógica vs Prolog

Escribir en fichero **'raz2.pl'.**
Formalizar en L. predicados y en Prolog
Ejecutar las preguntas

1: Carlos es alum.

2: Para que un sujeto sea alum es necesario que tenga buen tipo y

3: ésta es una condición suficiente para que esté macizo.

4: Si un sujeto no es alum, es atractivo.

5: Si un sujeto es atractivo, está macizo.

¿Carlos está macizo? ¿tú estás macizo?

El mundo de Pl-Man

Instalación de Plman http://lógica.i3a.ua.es

1. **Descargar** *pl-man.zip* (p. ej. en ~/Escritorio/)
2. **Abrir un terminal** y entrar donde está pl-man.zip .... $ **cd Escritorio**
3. **Descomprimir** *pl-man.zip* $ **unzip pl-man.zip**
4. **Entrar** en la carpeta plman/ $ **cd plman**
5. Comprobar que está todo: $ **ls**
   - **docs/** - manual de uso
   - **maps/** - mapas de ejemplo
   - **pl-man-game/** - código fuente del juego Plman
   - **plman** - script de lanzamiento.

Uso básico de Plman

1. Ver la ayuda del script $ **./plman --help**

2. Prototipo del script $ **./plman MAPA SOLUCION [PARAMETROS]**

Ejemplo $ **./plman maps/ejemplos/mapaej0.pl sol_mapaej0.pl**

3. Dentro de Pl-Man:
   - **ESC:** finalizar ejecución
   - Otra tecla: ejecutar un paso

Empecemos a resolver mapas…

maps/ejemplos/mapaej0.pl

1º **Edita** un fichero donde escribirás la solución al mapa (con extensión .pl):

$ gedit sol_mapaej0.pl &

2º En la **1ª línea escribe** (en todos los ficheros solución) y guarda

:- use_module('pl-man-game/main').

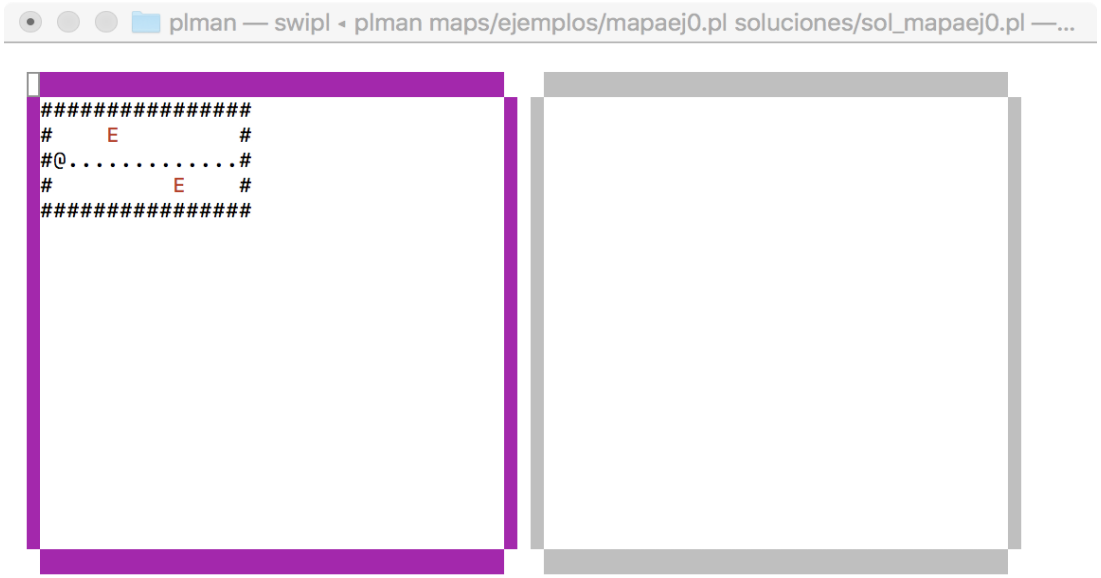
3º **Ejecuta** la solución (verás el mapa aunque no hayas escrito código):

…Escritorio **$ ./plman maps/ejemplos/mapaej0.pl sol_mapaej0.pl**

>> Suponemos que estamos en Escritorio/plman y ahí guardamos el fichero sol_mapaej0.pl. Si lo guardas en otro directorio debes escribir la ruta

maps/ejemplos/mapaej0.pl

¿Cómo hacemos que Plman se mueva?



ACCIONES de PLMAN

Plman puede realizar **1 acción en cada turno** (ciclo de ejecución con golpe espaciador), usando el predicado:

do(ACCION).

Donde **ACCION :**

move(X) : Plman se mueve en la dirección X

get(X) : Plman coge el objeto que se encuentra en la dirección X.

drop(X) : Plman deja el objeto en la dirección X

use(X) : Plman usa el objeto que lleva encima, en la dirección X

X ∈ D = { right, left, up, down }

>> Todas las acciones se hacen en el siguiente ciclo de ejecución.

>> Plman sólo puede llevar un objeto.

NORMAS para EJECUTAR las ACCIONES con ÉXITO

Plman puede:

- Moverse a una posición si en la misma **no hay** un objeto sólido.
- **Coger** un objeto si **no "lleva"** otro que haya cogido antes.
- **Dejar** el objeto si lo **lleva** "encima" y en la posición indicada para dejarlo no hay otro objeto.
- **Usar** el objeto si en la posición indicada es **factible su uso** (si lleva una llave podrá usarla en la dirección X si hay una puerta que la admite).
- Si debe usar **varios** objetos debe dejar el que lleva encima y coger el que necesite.
- **No moverse**, entonces hacer: **do(move(none))**

Ejemplos de cómo usar el predicado do/1 según la acción a realizar:

Si Plman quiere…

→ moverse a la derecha:	**do(move(right)).**
→ coger un objeto que se encuentra a su izquierda:	**do(get(left)).**
→ dejar un objeto arriba de donde él se encuentra:	**do(drop(up)).**
→ usar el objeto que lleva en la posición derecha:	**do(use(right)).**

Resuelve ... maps/ejemplos/**mapaej0**.pl

1º **Modifica** fichero solución añadiendo la(s) acción oportuna y **guarda**

:- use_module('pl-man-game/main').

Código ?????

2º **Ejecuta:**

…Escritorio $ **./plman maps/ejemplos/mapaej0.pl sol_mapaej0.pl**

Resuelve ...

maps/ejemplos/**mapaej1**.pl

1º **Crea fichero solución**

2º **Escribe regla y código** **:- use_module('pl-man-game/main').**

código…..?????

3º **Ejecuta:**

…Escritorio $ **./plman maps/ejemplos/mapaej1.pl tu_fichero.pl**

¿qué pasa cuando llega a la esquina?

Choca con la pared…mal asunto

SENSOR DE VISIÓN

Plman puede realizar diversas acciones **según lo que "vea"** a su alrededor.

Predicado predefinido: **see/3**

see(normal, DIR, OBJETO)

normal: visión de la posición **siguiente** a la ubicación de Plman

DIR = { right, left, down, up, here, down-right, down-left, up-right, up-left }

OBJETO: objeto que ve en la dirección DIR

Ejemplos

see(normal, right, ' ') → Éxito

see(normal, right, '.') → Fracaso

see(normal, left, 0) → Éxito

see(normal, down, X) → Éxito, instancia la variable X = 'o'

¿Cómo se mueve Plman según lo que ve?

>> Si ve un coco a la derecha entonces que se mueva a la derecha

Condicional

ver(right, '.') → mover(right)

Regla Prolog

do(move(right)) :- see(normal,right, '.')

>> En el cuerpo de una regla se pueden añadir varios predicados see/3 separados por comas (conjunción).

Ej. do(move(right)) :- see(normal, down, '.'), see(normal, up, ' ').

>> El predicado see/3 se puede negar:

Ej. not(see(normal,down,'E')) → tendrá éxito si no hay enemigo abajo.

Resuelve ...

maps/ejemplos/**mapaej1**.pl

1º **Modifica** fichero solución añadiendo las acciones oportunas y **guarda**

:- use_module('pl-man-game/main').

Código ?????

2º **Ejecuta:**

…Escritorio $ **./plman maps/ejemplos/mapaej1.pl tu_fichero.pl**

Prueba en mapaej1.pl

>> Cuando veas un coco entonces muévete en esa dirección

Condicional

$\forall x$ [ver(x, '.') $\rightarrow$ mover(x)]

Regla Prolog

do(move(DIR)) :- see(normal,DIR, '.')

*>> En cada regla sólo se ejecuta una **una acción**.*

*>> Para **cada turno** (golpe espaciador) se ejecuta una acción diferente.*

*>> **Se escriben tantas reglas** como **condiciones** puedan pasarle a Plman.*

*>> El **orden** en que se escriben las reglas es **MUY importante***

Prueba ...

maps/ejemplos/**mapaej3**.pl


```
:- use_module('pl-man-game/main').

do(move(DIR)) :- see(normal,DIR,'.').

do(get(DIR)) :- see(normal,DIR,'a').

do(use(DIR)) :- see(normal,DIR,'-').
```


PREDICADOS PREDEFINIDOS EN SWI-PROLOG

- Son predicados que ya están definidos en SWI-PROLOG y que tienen alguna funcionalidad asociada. Sólo pueden utilizarse en preguntas al intérprete o en el cuerpo de una regla.

- *Cuidado!* El usuario no puede utilizar el nombre de estos predicados como nombre de sus propios predicados porque no se pueden redefinir.

- Para conocer los predicados predefinidos de SWI-Prolog podemos utilizar la ayuda, poniendo ?- help. en el intérprete (swipl) o mirar el manual de SWI-Prolog.

Un predicado predefinido para escribir por pantalla es write/1 :

write(A): Escribe el átomo A por pantalla

```
write('Aquí empieza tu aventura').
```

writeln(A): Añade un retorno de carro al final

```
writeln('Aquí empieza tu aventura').
```

Cuando resuelvas un mapa es muy útil saber lo que “ está pasando”
Escríbete mensajes

```
:- use_module('pl-man-game/main').

do(move(left)) :- see(normal,left,'.'),
                  writeln('Me he comido un coco').
```