MATEMATICAS I

2019/2020

Sesión 2

INDEX

Repaso de Prolog

PLMan

y resolución de ejercicios

TERMINAL

swipl fichero.pl

“queries”

halt. (para salir)

PROLOG

predicado (P1, P2, ..., PN)

persona(matias).

coche_bueno(X):-coche(X),confortable(X),grande(X).

arreglo(X):-bueno(X),barato(X);rapido(X).

REPASO EJERCÍCIO "FAMILIA"

```
padre(juan, maria).
padre(matias, juan).

hijo(A,B) :- padre(B,A).
abuelo(A,B) :-
padre(A,C), padre(C,B).
```

```
primo(maria, luis).
primo(jose, maria).
primo(bea, luis).
```

EJERCÍCIO 1 - LA COSA NOSTRA

```
hermano(Hermano/a, Hermano/a)
```

EJERCÍCIO 2 - VIAJAR *GRAMER' STYLE*

Vamos de viaje y tenemos que hacer cuentas!

Crear un fichero llamado viaje1.pl

Colocar la siguiente información:

```
transporte(roma,20000).
transporte(londres,25000).
transporte(tunez,15000).

alojamiento(hotel,roma,25000).
alojamiento(hotel,londres,15000).
alojamiento(hotel,tunez,10000).
alojamiento(hostal,roma,15000).
alojamiento(hostal,londres,10000).
alojamiento(hostal,tunez,8000).
alojamiento(camping,roma,10000).
alojamiento(camping,londres,5000).
alojamiento(camping,tunez,5000).
```

EJERCÍCIO 2 - VIAJAR *GRAMER' STYLE*

Crear regla viaje/4 tal que:

```
?- viaje(roma,3,hotel,N).
N = 95000 ;
```

```
viaje (ciudad, días, alojamiento,
Resultado)
```

EJERCÍCIO 3 - *FAST AND NOT SO FURIOUS*

Conocimiento:

```
bueno('2cv').
barato('2cv').
barato(simca).
rapido(planet_express).
```

Predicados:

bueno_coche/1 -> bueno y barato o rapido

bueno_coche2/1-> bueno o rapido y barato

truno/1 -> no puede ser bueno

PLMAN

Descargar plman.zip

Abrir terminar e:

```
cd Descargas
unzip plman.zip
cd plman
ls
```

PLMAN - COMO USAR

Ver la ayuda: `./plman --help`

Prototipo: `./plman MAPA SOLUCION [PARAMETROS]`

```
./plman maps/ejemplos/mapaej0.pl sol_mapaej0.pl
```

Dentro de PLMan:

- **ESC**: finalizar ejecución
- **Espacio**: ejecutar un paso

PLMAN - COMO SOLUCIONAR

Recordar: PLMan sigue las reglas de Prolog

Ejemplo:

1. Crear fichero de solución: `gedit sol_mapaej0.pl &`

   (Remplazar gedit por editor favorito)
2. En la 1ª línea escribe (y en todos los ficheros solución)

   ```
   :- use_module('plman-game/main').
   ```
3. En la 2ª línea escribe y guarda

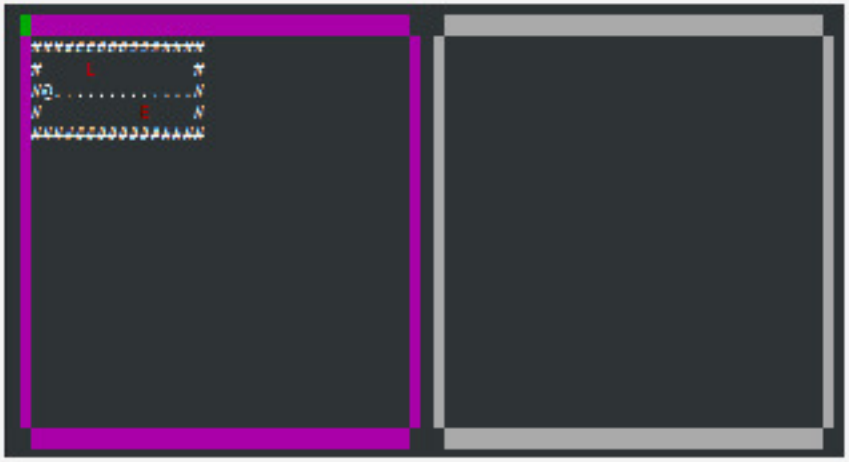
   ```
   do(move(up)).
   ```

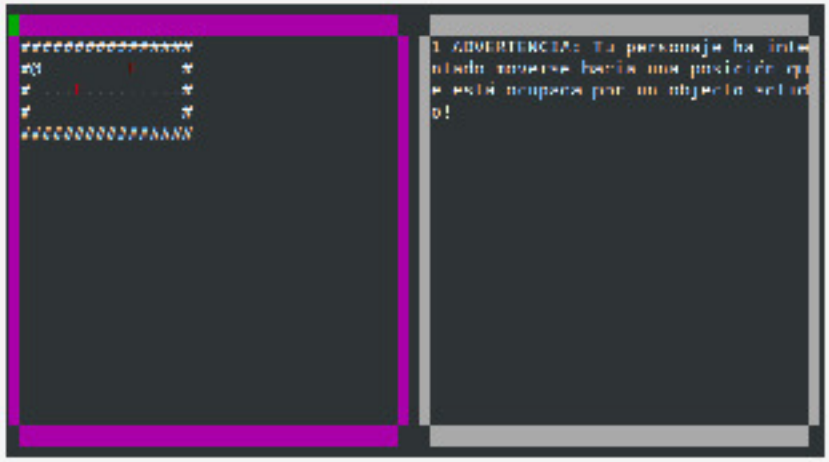
PLMAN - COMO SOLUCIONAR

Terminal

```
./plman maps/ejemplos/mapaej0.pl sol_mapaej0.pl
```

PLMAN - COMO SOLUCIONAR





```
==========================================
|              ESTADISTICAS DE EJECUCION
|
| * Estado final:        PACMAN HA MUERTO! :(
|
| + Cocos comidos:       0 / 13 (0%)
| + Movimientos:         0
| + Inferencias:         134268
| + Tiempo CPU:          0.189548390
|
| - Colisiones:          7
| - Intentos de accion:  0
| - Acciones erroneas:   0
| - Fallos de la regla:  0
==========================================
```

PLMAN - ¿COMO AVANZAR?

Plman puede realizar 1 acción en cada turno usando el predicado:

do(ACTION).

ACTION:

move(X)	Mueve en la dirección X
get(X)	Coge el objeto que se encuentra en la dirección X
drop(X)	Deja el objeto en la dirección X
use(X)	Usa el objeto que lleva encima, en la dirección X

[up,down,left,right]

PLMAN - ¿COMO AVANZAR?

ATENCIÓN

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

Plman sólo puede llevar un objeto.

PLMAN - ¿COMO AVANZAR?

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))

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)).

RESOLVER MAPA EJEMPLO 0

sol_mapaej0.pl

En que dirección están los cocos?

Como avanzar en esa dirección

RESOLVER MAPA EJEMPLO 1

maps/ejemplos/mapaej1.pl

No olvidar de:

```
:- use_module('plman-game/main').
```

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

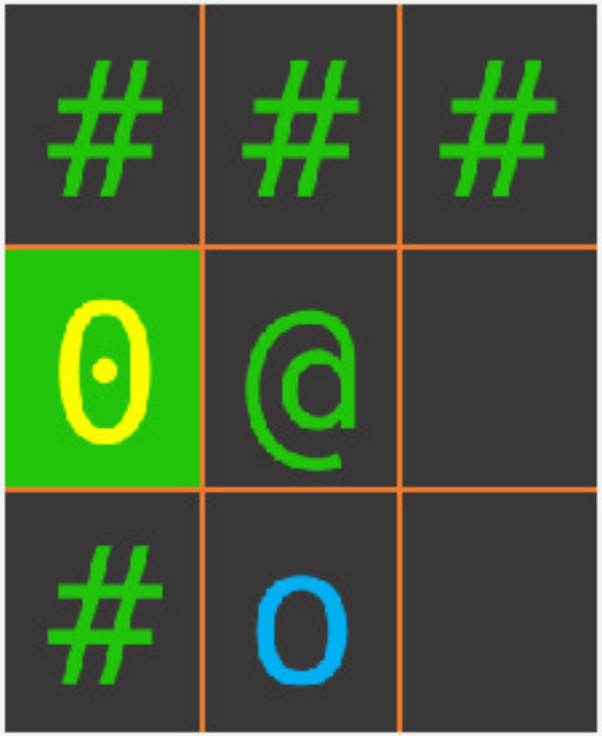
DIR = { right, left, down, up, here, down-right, down-left, up-right, up-left }

OBJETO: objeto que ve en la dirección DIR

SENSOR DE VISIÓN

Ejemplos

```
see(normal, right, ' ')  -> Éxito
see(normal, right, '.') -> Fracaso
see(normal, left, 0 ) -> Éxito
see(normal, down, X ) -> Éxito,
instancia la variable X = ‘o’
```



I SEE I MOVE

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).

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

El predicado see/3 se puede negar:

`not(see(normal,down,'E'))` -> **tendrá éxito si no hay enemigo abajo.**

RESOLVER MAPA EJEMPLO 1

Se puede resolver ahora el mapa?

Recuerda:

En cada regla sólo se ejecuta una acción.

Para cada turno se ejecuta una acción diferente.

Se escriben tantas reglas como condiciones.

El orden en que se escriben las reglas es MUY importante.

RESOLVER MAPA EJEMPLO 3

Intentar resolver el "maps/ejemplos/mapaej3.pl"

Ahora ha una llave que se tiene que coger y usar.

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

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