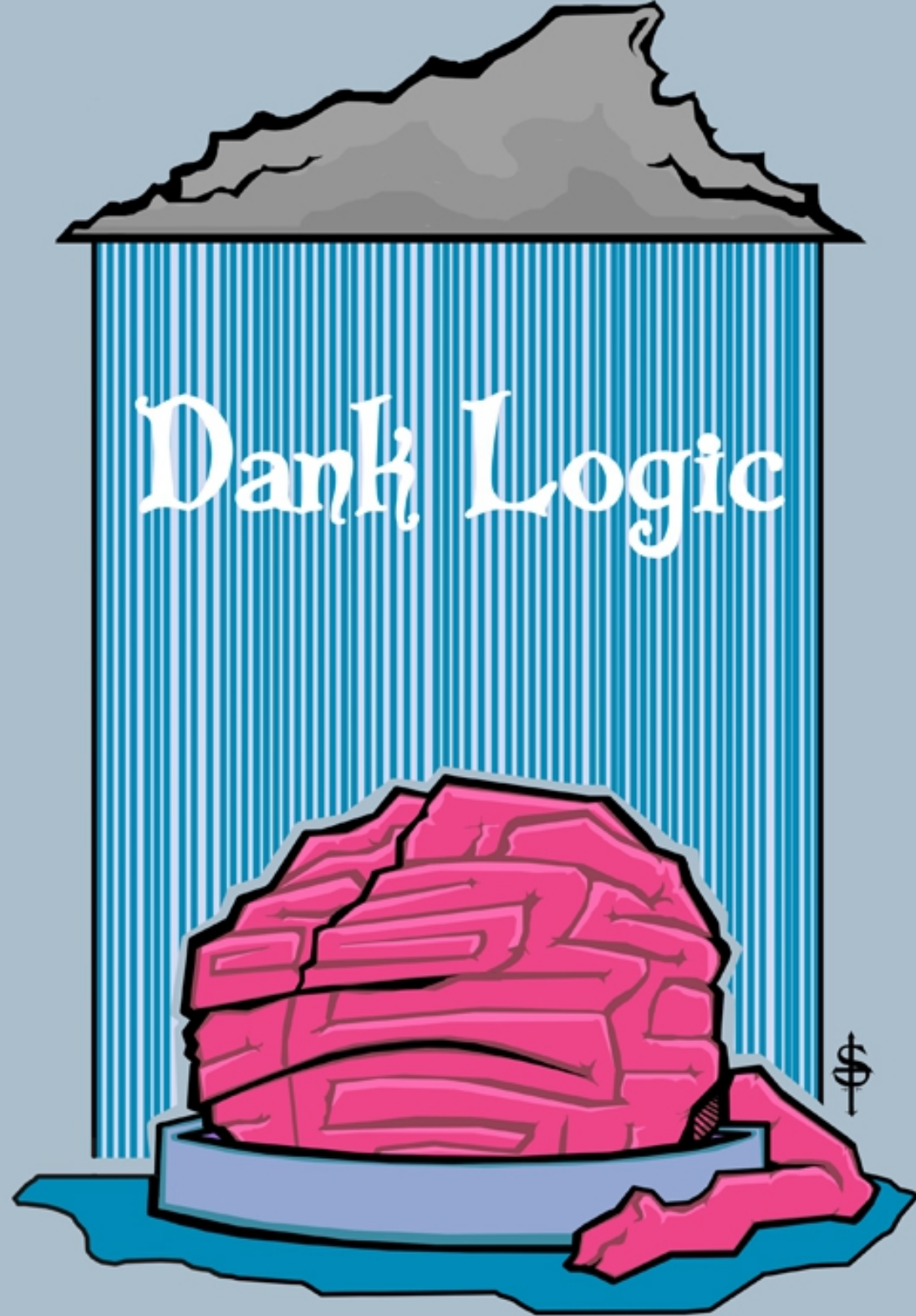


Prácticas de Lógica Computacional 2018/2019

Sesión 2

Dank Logic
by *saintpepsi





SWI Prolog

Repasando

Editor de textos

Find View Goto Tools Project Preferences Help

sesion1_ejemplo_nomansky.pl ruta_nivel1.txt

```
1 %%%%%%%%%%
2 %% HECHOS
3 %%%%%%%%%%
4
5 estrella(sol).
6 estrella(p_centauri).
7
8 planeta(tierra).
9 planeta(marte).
10 planeta(phobos).
11 planeta(proxima_b).
```

Intérprete

```
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 7.2.3)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- estrella(sol).
true.

?-
```

Preguntas

Respuestas

Salvar

Base de
Conocimientos



Consultar

USUARIO



Repasando comandos básicos

Ejemplos: <http://bit.ly/demosprolog>

TERMINAL

Lanzar intérprete	<code>\$ swipl</code>
Lanzar y consultar	<code>\$ swipl -f fichero.pl</code>
Lanzar editor	<code>\$ gedit</code>

INTÉRPRETE

Consultar Base	<code>?- consult('fichero.pl').</code>
Preguntar	<code>?- pregunta.</code>
Salir del intérprete	<code>?- halt.</code>

La base



Hechos

BASE DE CONOCIMIENTOS

```
estrella(sol).  
estrella(p_centauri).
```

```
planeta(tierra).  
planeta(marte).  
planeta(phobos).  
planeta(proxima_b).
```

INTÉRPRETE

```
?- estrella(sol).  
true.
```

```
?- planeta(pobos).  
false.
```

```
?- satellite(luna).  
ERROR: toplevel: Undefined procedure  
: satellite/1 (DWIM could not correct  
goal)
```

```
?- estrella(tierra).  
false.
```


```
?- █
```

Variables

BASE DE CONOCIMIENTOS

```
planeta(tierra).  
planeta(marte).  
planeta(phobos).  
planeta(proxima_b).  
  
orbita(tierra, sol).  
orbita(marte, sol).  
orbita(phobos, marte).  
orbita(proxima_b, p_centauri).
```

INTÉRPRETE

```
?- planeta(P).  
P = tierra .  
  
?- orbita(phobos, QUE).  
QUE = marte.  
  
?- orbita(H_H_H, sol).  
H_H_H = tierra ;  
H_H_H = marte.  
  
?- 
```

Conjunciones y Disyunciones

INTÉRPRETE

```
?- orbita(P, marte) ; orbita(P, sol).
```

```
P = fobos ;
```

```
P = tierra ;
```

```
P = marte.
```

```
?- orbita(P1, P2), planeta(P1), planeta(P2).
```

```
P1 = fobos,
```

```
P2 = marte ;
```

```
false.
```

```
?- █
```


Ejercicios

Descargar **sesion2_nomansky_hechos.pl** (Campus)

Resolver preguntando

- ¿Existe el planeta **miranda**?
- ¿Alrededor de quién orbita?
- ¿Cuántos planetas orbitan **urano**?
- ¿Dónde está el arma llamada **saterant**?
- ¿Hay seres vivos en **pluton**? ¿Cuántos?
- ¿Hay jugadores en los satélites de **jupiter**?
- Nombra los planetas que orbitan **haumea** o **eris**.
- En algún satélite de **neptuno** hay un objeto que también está en **urano**, ¿Cómo se llama?

Predicado

Forma de relacionar datos (términos u otros predicados)

predicado(arg1, arg2, arg3, ..., argN)

- Empiezan siempre por letra **minúscula**.
- No hay espacio entre el nombre y el paréntesis.
- Argumentos separados por comas.
- Pueden tener 0 argumentos (y no habría paréntesis)

Ejemplos

% habitacion(X): X es una habitación
habitacion(salon).

% puntos(J, P): El jugador J tiene P puntos
puntos(juan, 150).

% telefono(P, N, C): Persona P tiene número
% de teléfono N, de la compañía C
telefono(miguel, '55535261', hablomovil).

% encima_de(O1, O2): El objeto O2 está encima
% del objeto O1
encima_de(mueble(mesa), juguete(pelota)).

Términos: átomos y números

TÉRMINO

Cualquier número, átomo o variable.

ÁTOMO (constante)

Palabra para identificar a un objeto del dominio.

- **Siempre** empieza por **letra minúscula**.
- Puede contener letras, números o subrayado.
- No pueden contener espacios.
- Si va entre comillas simples " puede contener cualquier cosa.

Ejemplos

```
% objeto(X): X es un objeto  
% Átomos: piedra, 'escopeta recortada'  
%      lampara_magica, entrada512  
objeto(piedra).  
objeto('escopeta recortada').  
objeto(lampara_magica).  
objeto(entrada512).
```

```
% numero(N): N es un número  
% Números: 123, 12.5, -10  
numero(123).  
numero(12.5).  
numero(-10).
```

Comentarios

Los comentarios son ignorados por el intérprete de Prolog. Sirven para hacer el código más fácil de entender.

```
%  
% Predicado principal para dibujar el árbol  
%  
xmas(N):- N >= 1, SpcIni is N + 3,  
          spcs(SpcIni), write('/\\\\n'),  
          dibujaArbol(N, SpcIni, 1).
```

Términos: variables

VARIABLE

Representan **un término cualquiera** del dominio.

- Empiezan por **letra mayúscula o subrayado**.
- Pueden contener letras, números o subrayado.
- No pueden contener espacios.

Las variables pueden estar:

- **Sin instanciar**: aún no tienen valor.
- **Instanciadas**: están asociadas a un término.

Ejemplos

% Predicados definidos antes

% Cualquier objeto
objeto(X).

% Cualquier habitación
habitacion(_unaHabitacion).

% PNTS = Los puntos que tiene juan
puntos(juan, PNTS).

Hechos

HECHO

Una cláusula concreta de un predicado. Son **datos** que se añaden directamente a la Base de Conocimientos.

% Estadísticas de los personajes

% personaje(NOMBRE, FUERZA, MANÁ)

personaje(guerrero, 80, 40).

personaje(mago, 35, 95).

personaje(elfo, 65, 60).

% Salas especiales con tesoros

% sala(SALA, NUM_TESOROS)

sala(pasillo_3, 5).

sala(catacumba_sur, 8).

Reglas

CABEZA : - CUERPO.

Cabeza: puede ser cierto o falso, dependiendo del CUERPO.
Es un dato que depende de otros.

Cuerpo: condiciones que deben cumplirse.

CABEZA es cierto si son ciertas las condiciones del CUERPO.

A la CABEZA se le suele llamar **objetivo**, y se dice que tiene éxito cuando es cierto, o que fracasa en caso contrario.

Ejemplo

%%%% HECHOS

% Personajes

personaje(guerrero).

personaje(mago).

% Objetos del inventario de cada personaje

objeto(guerrero, espada).

objeto(guerrero, escudo).

objeto(mago, libro_sagrado).

%%%% REGLAS

mortal(guerrero) :- personaje(guerrero).

mortal(P) :- personaje(P).

puede_cubrirse(P) :- objeto(P, escudo).

Predicados predefinidos

PREDICADO PREDEFINIDO

Un predicado que ya viene definido en SWI-PROLOG y que tiene alguna funcionalidad asociada. Sólo pueden utilizarse en preguntas al intérprete o **en el cuerpo de una regla**.

Si ponemos hechos de un predicado predefinido en un programa nuestro dará error al compilar, porque no podemos redefinirlo.

*Para conocer los predicados predefinidos de swi-prolog podemos utilizar la ayuda, poniendo **?- help.** en el intérprete.*

Ejemplos

Para escribir por pantalla

% write(A): Escribe el átomo A por pantalla

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

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

% get_single_char(C): Espera a que el usuario

% pulse una tecla y unifica el valor de la tecla

% pulsada con C

% listing: lista el programa prolog en memoria

% listing(C): lista sólo las cláusulas

% del predicado C, dentro del programa.

% halt: Cierra el intérprete de SWI-PROLOG.

Conectivas lógicas

Permiten describir condiciones complejas en reglas o preguntas al intérprete. Las conectivas fundamentales son el conjuntor (,), el disyuntor (;) y el negador(not(X)).

% Ejemplo de reglas con conjuntor y disyuntor

```
novios(luis, maria) :-  
    quiere(luis, maria), quiere(maria, luis).
```

```
resultado(luis, perder) :-  
    not(mas_puntos(luis, maquina)) ;  
    abandona(luis).
```

Añadiendo Reglas

Añadir las siguientes reglas a la Base de Conocimientos

% es_planeta(**P**): P orbita alrededor de una estrella
es_planeta(P) :- ...

% es_satelite(**S**): S orbita alrededor de un planeta
es_satelite(P) :- ...

% arma_planetaria(**A**): A es un arma que está en un
% planeta (no en un satélite).
arma_planetaria(A) :- ...

% ser_satelital(S): S es un ser_vivo que habita en
% un satélite
ser_satelital(S) :- ...

Complicándolo un poco

```
% planeta_rico(P): P es un planeta rico si tiene  
% objetos, naves y armas  
planeta_rico(P) :- ...
```

```
% satellite_vivo(P): P es un satellite vivo si tiene  
% al menos un jugador y un ser_vivo  
satellite_vivo(P) :- ...
```

```
% puede_viajar(J): J puede viajar si es un jugador  
% y hay una nave dónde está  
puede_viajar(J) :- ...
```

```
% alcanza_objeto(J, O): El jugador J alcanza el  
% objeto O si ambos están en el mismo lugar  
alcanza_objeto(J, O) :- ...
```


Más complicado aún

% teleport(**P1, P2**): Es posible teleportarse de P1
% a P2 si en ambos hay una nave con el mismo nombre
teleport(P1, P2) :- ...

% **planetas_hermanos(P1, P2)**: P1 y P2 son planetas
% hermanos si hay un ser vivo que se llame igual
% en alguno de sus respectivos satélites
planetas_hermanos(P1, P2) :- ...

% objeto_sagrado(**O**): Es un objeto que sólo se
% encuentra en un planeta que tiene planeta hermano
% (no se encuentra en ningún otro planeta).
objeto_sagrado(O) :- ...

Fuentes para aprender PROLOG

TUTORIALES Y APUNTES

Adventure in prolog (inglés)

<http://www.amzi.com/AdventureInProlog/advfrtop.htm>

Apuntes de Prolog y material (castellano)

<http://www.dccia.ua.es/logica/prolog/docs/prolog.pdf>

<http://www.dccia.ua.es/logica/prolog/material.htm>

LIBROS

The Art of Prolog

<http://gaudi.ua.es/uhtbin/cgisirsi/?ps=gIXvHaO0MD/0/274340092/9>
ejemplos

Programación en Prolog

<http://gaudi.ua.es/uhtbin/cgisirsi/?ps=5h8XgkxL2D/x/185850102/9>

Ejercicios