MODELOS FORMALES DE COMPUTACIÓN
Introducción a Prolog

---

1. SWI-Prolog

SWI-Prolog es de libre distribución y está disponible para Windows, Linux y MacOs. En los PC's de los laboratorios del DSIC, SWI-Prolog opera sobre Windows. Pueden obtenerse ésta y otras versiones de SWI-Prolog vía WWW en la siguiente dirección de internet:

http://www.swi-prolog.org

El sistema está accesible en el menú de Inicio → Programas → SWI-Prolog → Prolog. Una vez abierto aparece la ventana del entorno en cuya parte central aparece el mensaje

```
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 5.8.0)
Copyright (c) 1990-2009 University of 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).

?-
```

El sistema ejecuta un intérprete que queda a la espera de que introduzcamos consultas a la base de conocimiento (programa Prolog) cargada en memoria. En SWI-Prolog estas consultas pueden realizarse haciendo uso de un shell interactivo cuyo prompt es "`?-`". Desde esta línea de entrada de consultas, se pueden cargar programas guardados en un fichero. Para cargar un programa podemos usar las opciones del menú `File/Consult` o bien introducir directamente la orden `consult(`nombrefichero`)` o `[`nombrefichero`]` seguidos de un punto. Podemos salir del intérprete mediante el comando `halt.` o bien pulsando CTRL+D.

Ejercicio 1 *Edita el siguiente programa Prolog, almacenándolo en el fichero* `familia.pl`

```
padrede(juan, maria).
padrede(pablo, juan).
padrede(pablo, marcela).
padrede(carlos, debora).
padrede(luisa, debora).
hijode(B, A) :- padrede(A, B).
hermanode(A, C) :- padrede(B, A), padrede(B, C), A\==C.
abuelode(A, C) :- padrede(A, B), padrede(B, C).
familiarde(A, B) :- padrede(A, B).
familiarde(A, B) :- hijode(A, B).
familiarde(A, B) :- hermanode(A, B).
familiarde(A,B) :- abuelode(A,B).
```

Carga el programa guardado en el fichero `familia.pl` *haciendo uso de alguna de las instrucciones de carga anteriores. NO olvides que todas las consultas finalizan con un punto.*

Si no muestra ningún mensaje de error, el fichero está cargado correctamente.

El predicado `padrede` de aridad 2 expresa que su primer argumento está relacionado con el segundo mediante la relación *es padre de*. La definición de este predicado es extensional, es decir, en el programa aparecen explícitamente los hechos de la relación. El predicado `hijode(A,B)` se define intensionalmente mediante una regla y expresa que `A` es hijo de `B` *si* `B` es padre de `A`. Es decir, podemos saber quién es hijo de alguien consultando la relación *es padre de* definida por el predicado `padrede`. Fíjate que el *si* condicional de la frase anterior se expresa en el programa con el par de símbolos `:-`. La definición del predicado `hermanode` contiene tres predicados separados por comas en el cuerpo de la regla (detrás de `:-`). Las comas representan una conjunción de predicados. Es decir, al ser consultados los predicados en la base de conocimiento, el intérprete debe devolver una respuesta de éxito para todos ellos. La definición del predicado `abuelode` expresa la relación entre pares formados por padres e hijos. En la definición del predicado `familiarde` de aridad 2 puedes observar que, además de ser intensional, se usan tres cláusulas en su definición. Ésta es una forma simple de expresar la disyunción en Prolog. En este caso, se define que el primer argumento `A` es familiar del segundo argumento `B` *si* `A` es padre, hijo, o hermano de `B`.

2. Consultas a la base de conocimiento

A continuación vamos a realizar varias consultas sobre la base de conocimiento cargada en memoria. Podemos lanzar consultas desde el intérprete. Empezamos realizando consultas sencillas sobre los hechos del programa. Por ejemplo, preguntar al intérprete si Juan es padre de María, se escribiría así, donde `juan` y `maria` no son variables sino constantes

```
?- padrede(juan,maria).
true.
```

Si quisiéramos ser más precisos, lo que hemos hecho es preguntar al intérprete si en la base de conocimiento se encuentra algún hecho que pruebe que Juan es padre de María.

En las consultas se pueden usar variables para extraer más información de la base de cono- cimiento. Por ejemplo, supón que queremos saber si con el conocimiento almacenado en nuestro programa María tiene un padre, y en tal caso, saber de quién se trata. Para ello podemos lanzar al intérprete la misma consulta que antes, pero con el primer argumento como una variable. Como respuesta se obtiene el valor de la variable con el que nuestra consulta es cierta.

```
?- padrede(X,maria).
X = juan.
```

Es decir, que el intérprete puede demostrar que `padrede(X,maria)` es cierto a partir de los hechos del programa si `X=juan`.

En el caso de que exista más de una respuesta afirmativa a una consulta, el sistema retorna la primera que computa. Si deseamos recibir más respuestas debemos pulsar `;` o la tecla `r` (*redo*) para conocer otras respuestas. Compruébalo realizando el siguiente ejercicio.

Ejercicio 2 *¿Qué ocurre si preguntas quiénes son los hijos de Pablo? Para ello reescribe la consulta del ejemplo anterior cambiando la variable* `X` *por el término* `pablo` *y el término* `maria` *por una variable (por ejemplo* `X`*). ¿Qué ocurre cuando pulsas la tecla* `r` *después de la primera respuesta?*

Hasta este punto, hemos realizado consultas que sólo requerían buscar si un par de términos (juan, maria, pablo. . .) están relacionados entre sí por un hecho (`padrede`). En el caso de usar variables, éstas se instanciaban con los términos que hacían el hecho cierto. Ahora vamos a realizar consultas que requieren `deducir` información que no está expresada con hechos en el programa, pero que puede obtenerse a partir de los hechos y las reglas. Las reglas son del tipo "Si es verdad el antecedente (cuerpo) entonces es verdad el consecuente (cabeza)". Por ejemplo: haciendo uso de la regla `hijode(A,B) :- padrede(B,A).` que expresa que si `B` es padre de `A` entonces `A` es hijo de `B`, el intérprete puede llegar a la conclusión de que María es hija de Juan tras la siguiente consulta:

```
?- hijode(maria,juan).
true.
```

Ejercicio 3 *Haciendo uso de* `familia.pl`*:*

Consultar quién es abuelo de María.

Consultar todos los familiares de Pablo conocidos.

Consultar todos las personas que son familia de Débora.

Usando un editor de texto, añadir al fichero `familia.pl` *una regla que defina la relación* es nieto de *similar a* es hijo de *pero usando el predicado* `abuelode` *en lugar de* `padrede`. *Guarda los cambios y recarga el fichero con el predicado* `reconsult(familia).` *Averigua de quién es nieta María usando el predicado* `nietode`.

Lanza la consulta `familarde(A,B).` *Añade al programa la regla* `familarde(A,B) :- nietode(A,B)` *y reconsultalo. ¿En qué difieren las dos respuestas computadas?*

3. Practicando con Prolog

Realiza los siguientes ejercicios para practicar con el Prolog:

Ejercicio 4 La unificación el mecanismo que permite dar valor a las variables en los objetivos. Para que dos predicados unifiquen han de tener el mismo nombre, la misma aridad y deben unificar cada uno de los términos de sus parámetros. En Prolog el operador = representa la unificación. Comprueba si unifican o no los siguientes términos o átomos usando el intérprete de Prolog:

`padrede(X,debora)` y `padrede(carlos,debora)`

`X` y `fecha(10,nov,Y)`

`fecha(10,oct,2013)` y `fecha(X,nov,2013)`

`momento(fecha(10,nov,2013),Y)` y `momento(X,hora(13,05))`

Ejercicio 5 Edita el siguiente programa Prolog y almacenálo en el fichero `fiesta.pl`

```
hombre(alfredo).
hombre(felipe).
hombre(francisco).
mujer(sonia).
mujer(eva).
mujer(carmen).
bebe(alfredo, whisky).
```

```
bebe(alfredo, ron_cola).
bebe(felipe, cerveza).
bebe(felipe, gin_tonic).
bebe(felipe,ron_cola).
bebe(francisco, vino).
bebe(francisco, malibu).
bebe(sonia, gin_tonic).
bebe(sonia, malibu).
bebe(eva, vino).
bebe(eva, cerveza).
bebe(carmen, whisky).
bebe(carmen, ron_cola).
```

a) Define un predicado `pareja(X,Y)` que tenga éxito cuando `X` es un hombre e `Y` una mujer y tengan al menos una bebida favorita en común. Averigua qué parejas tienen `Alfredo` y `Francisco`.

b) Modifica el programa para reflejar los siguientes hábitos de bebida:

`Pepe` bebe cualquier cosa que beba `Alfredo`.

`Elena` bebe cualquier cosa que beban `Sonia` o `Felipe`.

c) Averigua qué parejas tienen `Pepe` y `Elena`.

Ejercicio 6 Sea el siguiente enunciado:

```
Bertoldo y Bartolo son rufianes.
Romeo y Bertoldo, como su nombre indica, son nobles.
Bartolo es un plebeyo.
Gertrudis y Julieta son damas.
Julieta es hermosa.
Los plebeyos desean a cualquier dama, mientras que los nobles solo a aquellas que son
hermosas.
Los rufianes, para satisfacer sus instintos, raptan a las personas a las que
desean.
```

Representa el enunciado como un programa Prolog. Responde a las siguientes preguntas:

¿Qué noble es un rufián?

¿A quién podría raptar `Romeo`?

¿Quién puede raptar a `Julieta`?

¿Quién rapta a quién?

¿A quién desea `Bartolo`?

¿Y `Romeo`?

¿Qué hermosa dama es deseada por `Bartolo`?

Ejercicio 7 Edita el siguiente programa Prolog y almacénalo en el fichero `menu.pl`

```
primero(ensalada).
primero(sopa).
primero(revuelto).
segundo(chuleta).
segundo(lubina).
segundo(pollo).
postre(natillas).
postre(flan).
```

```
postre(cafe).
comida(X, Y, Z):- primero(X), segundo(Y), postre(Z).
```

Este programa describe la información con la que un restaurante confecciona el menú del día, el cual está formado por un primero, un segundo y postre. Utiliza el predicado `comida/3` para determinar las distintas comidas que puede servir el restaurante. ¿Puedes predecir el orden en que se obtienen estas comidas? ¿De qué depende este orden?