DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Sistemas Inteligentes de Gestión

Relación de ejercicios

PROLOG

© *Juan Carlos Cubero & Fernando Berzal*

ENTREGA DE LA PRÁCTICA

```
1_ejemplos.txt
2_ejemplos.pl
2_ejemplos.txt
3_ejemplos.txt
4_ejemplos.pl
5_genealogia.pl
6_maximo.pl
7_repeticiones.pl
8_borrado.pl
9_inserta.pl
10_herencia.pl
11_sin_repeticiones.pl
12_mezcla.pl
13_partir.pl
```

Ejercicios tipo C

Ejercicio 1

Cree un fichero de texto llamado `1_ejemplos.txt`.

Suponiendo que el predicado `gusta(X,Y)` indica que a `X` le gusta `Y`, escriba los objetivos que habría que definir en Prolog para contestar las siguientes preguntas:

- ¿Le gusta algo a Juan?
- ¿Le gustan a Juan las almendras?
- ¿Qué es lo que le gusta a Juan?

A continuación, suponiendo que `esPadre(X,Y)` indica que `X` es el padre de `Y`, exprese en lenguaje natural lo que significan las preguntas:

```
?- esPadre(X,pedro).

?- esPadre(_,pedro).

?- esPadre(X,_).

?- esPadre(_,_).

?- esPadre(_,X).

?- esPadre(X,Y).

?- esPadre(personaQueNoExiste,pedro).
```

Ejercicio 2

Represente, primero en Lógica de Predicados y luego en Prolog, los siguientes hechos:

- Pedro quiere a María.
- Pedro quiere a Belén.
- Manuel quiere a Belén.
- María quiere a Pedro.
- Todos quieren a Juan y a María.
- Todo el mundo se quiere a sí mismo.
- Alguien quiere a Juan y a Pedro.
- Alguien quiere a María.
- Alguien quiere a todos los demás.

Almacénelos en un fichero con nombre `2_ejemplos.pl`

A continuación, escriba los objetivos en Prolog necesarios para responder a las preguntas indicadas a continuación:

- ¿Quiere Manuel a María?
- ¿Quiere Manuel a María y a Pedro?
- ¿Quiere Manuel a alguien?
- ¿Quiénes son los que quieren a María?
- ¿Quiénes son los que quieren a Pedro?
- ¿Quiénes son los que se quieren mutuamente?
- ¿Quiénes son los que se quieren a sí mismos?
- ¿Se quiere Manuel a sí mismo?
- ¿Hay alguien que quiera a María?
- ¿Hay alguien que quiera a alguien?
- ¿Hay alguien que se quiera a sí mismo?
- ¿Hay alguien que quiera a todo el mundo?

Incluya las respuestas proporcionadas por Prolog en el fichero `2_ejemplos.txt`. En el caso de que no se pueda plantear alguna de las preguntas, indique el motivo.

Ejercicio 3

Escriba en Lógica de Predicados los hechos y reglas necesarios para representar:

- "Existe una persona que gobierna a todos los guatemaltecos".
- "Todos los guatemaltecos tienen un animal".
- Reglas que relacionen `TienePerro` y `TieneGato` con `TieneAnimal`.
- "Cada dueño debe vacunar de la rabia a su(s) perro(s)".

Utilice los predicados `Gobierna`, `EsGuatemalteco`, `TieneAnimal`, `TienePerro`, `TieneGato`, `EsPerro`, `EsVacuna`, `EsEnfermedad`, `DebeVacunar`…

Guarde su solución en un fichero llamado `3_ejemplos.txt`

Ejercicio 4

Traduzca a Prolog los predicados del ejercicio anterior y guarde su solución en el fichero `4_ejemplos.pl`.

> RECORDATORIO: Si una variable aparece una sola vez en el ámbito de una regla, tendrá que usar el símbolo _ para representarla (tal y como se hace para especificar objetivos en línea de comandos).

Para cada uno de los siguientes ejercicios, incluya un breve comentario acerca de cómo funciona cada regla que defina. Dicho comentario se incluirá justo antes de la definición en Prolog de la regla correspondiente, dentro del fichero `.pl` *asociado al ejercicio.*

Ejercicio 5

En un fichero llamado `5_genealogia.pl`, cree las reglas en Prolog necesarias para representar un árbol genealógico utilizando únicamente el predicado

`padres(<nombre padre>,<nombre madre>,<nombre hijo>)`

Incluya datos de su propio árbol genealógico (incluyendo, al menos, datos de abuelos, padres, nietos, y bisnietos).

a. Indique el objetivo que permite responder la pregunta ¿Quién es padre de quien?

b. Incluya una regla que represente el parentesco de hermandad.

c. ¿Puede aumentarse la potencia del predicado anterior con una regla que nos diga explícitamente que la propiedad de ser hermanos es conmutativa (es decir, que si X es hermano de Y, entonces Y es hermano de X)?

d. Incluya una regla (o varias) que represente los abuelos paternos y maternos. Establezca los objetivos necesarios para responder las siguientes preguntas:
   - ¿Cuáles son todos los abuelos de cada persona?
   - ¿Cuáles son todos los nietos de cada persona?

e. ¿Cuáles son los ascendientes de una persona? (se necesita recursividad).

f. Cambie las reglas para obtener los parentescos de una forma eficiente, detectando cuáles son los argumentos que se pasan variables (se necesita usar el predicado `var`).

Ejercicio 6

Escriba un programa en Prolog 6 para calculad el mayor elemento de una lista de enteros. Incluya las reglas necesarias y los hechos utilizados como batería de pruebas en el fichero `6_maximo.pl`.

Ejercicios tipo B

Ejercicio 7

Escriba un programa que cuente el número de veces que un elemento se encuentra repetido en una lista (`7_repeticiones.pl`).

Ejercicio 8

Escriba un programa en Prolog que borrad un elemento de una lista (`8_borrado.pl`).

El resultado se guardará en otra lista nueva. Por ejemplo, el resultado de borrar el valor `a` de la lista `[b,a,g,a,h,b]`, sería la nueva lista formada por `[b,g,h,b]`.

Ejercicio 9

Escriba un programa en Prolog que insertad de forma ordenada un entero en una lista ya ordenada de enteros (`9_inserta.pl`)

> NOTA: Guarde el resultado en una segunda lista.

Ejercicios tipo A

Ejercicio 10

Cree un fichero `10_herencia.pl` con hechos del siguiente tipo:

```
es_un(rapaz,ave).
es_un(aguila,rapaz).
es_un(halcon,rapaz).
es_un(buitre,rapaz).
es_un('aguila perdicera',aguila).
es_un(calzada,aguila).
es_un('cernicalo primilla',halcon).
es_un('cernicalo vulgar',halcon).

tiene(ave,plumas).
tiene(ave,pico).
tiene(rapaz,garrasFuertes).
tiene(rapaz,vistaProfunda).
```

El predicado `es_un(C1,C2)` se utiliza para representar que la clase de objetos `C1` es una subclase directa de `C2`. El predicado `tiene(C,P)` se usa para representar que los objetos de la clase `C` admiten la propiedad `P`.

a. Aumente la potencia del predicado `tiene`, añadiendo una regla (con cabecera `tiene`) que diga que las propiedades aplicables a una clase también son aplicables a cualquier subclase que herede de ella (directamente o a través de una línea de herencia), para lo que tendrá que usar el predicado `es_un`.

   Por ejemplo, podremos deducir que un cernícalo primilla tiene alas; es decir, `tiene('cernicalo primilla',alas)` debe devolver un éxito.

b. Añada las reglas necesarias para poder definir el predicado `es_un_tipo_de(C1,C2)`, que nos dirá si la clase `C1` es una subclase directa de `C2` o existe una línea de herencia entre `C1` y `C2`.

   Por ejemplo, podremos deducir que un cernícalo primilla es un tipo de ave; es decir, `es_un_tipo_de('cernicalo primilla',ave)` debe devolver éxito.

c. Nos preguntamos ahora si no hubiese sido mejor definir la regla con cabecera tiene del apartado a), usando el predicado `es_un_tipo_de` definido en el apartado b), en vez de utilizar el predicado `es_un`. Justifique su respuesta.

Ejercicio 11

Escriba un programa en Prolog que quite las repeticiones de elementos de una lista, guardando el resultado en una segunda lista (`11_sin_repeticiones.pl`).

Por ejemplo, `[a,b,h,j]` se obtendría como resultado de eliminar los elementos repetidos de la lista `[a,b,h,b,a,j]`.

Ejercicio 12

Defina un predicado llamado `mezclaOrdenada` para mezclar dos listas ordenadas de enteros sin repetidos en una tercera, también ordenada y sin repetidos (`12_mezcla.pl`).

Por ejemplo, el resultado de mezclar las listas `[1,3,5]` y `[2,3,9]` sería la lista `[1,2,3,5,9]`.

Se pueden definir varias reglas, pero todas deben tener la misma cabecera. No se pueden usar reglas auxiliares con otra cabecera.

Ejercicio 13

Defina un predicado llamado `partir` para dividir una lista respecto un umbral, dejando los valores menores a la izquierda y los mayores a la derecha (`13_partir.pl`).

Por ejemplo, el resultado de partir la lista `[2,7,4,8,9,1]` respecto al umbral `6` serían las listas `[2,4,1]` y `[7,8,9]`.

EVALUACIÓN DE LAS PRÁCTICAS

Para cada problema, se crearán los correspondientes ficheros `.pl` con la solución del ejercicio correctamente implementada en Prolog.