Programación Lógica (1ª parte)

3.1. Introducción

La idea fundamental de la programación lógica es emplear la Lógica como lenguaje de programación que nos permite resolver problemas en los que intervienen objetos y relaciones entre ellos.

La principal característica de la programación lógica es que se especifica qué se tiene que hacer, es decir, se describe el problema a resolver y las condiciones que han de satisfacer sus soluciones, y no cómo se debe hacer. Por tal motivo se dice que la programación lógica es declarativa frente a la programación "tradicional" que es imperativa.

Por tanto, para escribir un programa lógico debemos identificar qué objetos intervienen en el problema y cuáles son las relaciones entre éstos. Una vez identificados estos elementos, tal y como describimos en las secciones siguientes, debemos representar los objetos mediante términos y definir las relaciones entre ellos mediante hechos y reglas.

3.2. Algunas consideraciones sobre la sintaxis de Prolog

Como sabemos, un término de la Lógica de predicados puede ser un símbolo de constante, un símbolo de variable o un símbolo de función aplicado a tantos términos como indica sus aridad. Estos elementos se incluyen en Prolog de la forma siguiente.

- Los símbolos de variable, o simplemente variables, son semejantes a las incógnitas y designan objetos indeterminados. Una variable se representa por una secuencia de letras, dígitos y el símbolo _ comenzando siempre con mayúscula ó con el símbolo _. Ejemplos de nombre de variable son:

  X1, Padre, _X, Numero_de_Telefono, Calle32.

- Los símbolos de constante representan objetos concretos mediante un nombre. Se pueden escribir como una secuencia de letras, dígitos y el símbolo _ comenzando siempre con minúscula, o bien como una cadena de caracteres encerrada entre comillas simples. Son símbolos de constante en Prolog:

  antonio, x, lista_vacía, nil, x25, 'pepito grillo', 'Antonio', '123 456'.

- Los símbolos de función permiten designar nuevos objetos a partir de otros. Se representan igual que los símbolos de constante por lo que su primera letra debe ser una minúscula. Ejemplos de términos en Prolog que se construyen con símbolos de funciones son:

  padre(pepe), padre(Pepe), suma(X,cero), color(casa_de(X)).

Nótese que en el primer ejemplo anterior, pepe es un símbolo de constante, mientras que en el segundo ejemplo, Pepe es un símbolo de variable. En el tercer ejemplo, X representa una variable mientras que cero una constante. En el cuarto ejemplo, tanto color como casa_de los consideramos símbolos de función.

Realmente, el concepto de término en Prolog es más amplio que el de la Lógica de predicados, pues además de los mencionados anteriormente, se incluyen en esta categoría a los números enteros y reales como constantes.

En Prolog a los símbolos de constante se les denomina átomos. Un símbolo de función junto con su aridad se denomina funtor, lo que posibilita que pueda emplearse el mismo nombre de función con diferentes aridades, pues para Prolog se consideran funtores diferentes. Una estructura es el resultado de aplicar un funtor a tantos términos como indica su aridad.

Los símbolos de predicado se utilizan para expresar propiedades de los objetos (predicados monádicos) y relaciones entre ellos (predicados poliádicos). En Prolog, los símbolos de predicado se representan igual que los símbolos de función, siendo el contexto quien nos permite distinguir entre unos y otros.

Como veremos con más detalle en la próxima práctica, las fórmulas de la Lógica de predicados que se representan en Prolog son un tipo especial de sentencias que se escriben con todos sus cuantificadores en la parte de la izquierda. Tales cuantificadores, si los hay, serán todos universales, motivo por el cual se omiten a la hora de escribirlos. Dicho de otra forma, todo símbolo de variable se supone que está cuantificado universalmente, y así no se escribe el cuantificador correspondiente.

Por tanto una fórmula de la Lógica de predicados como la siguiente,

$$\forall x \forall y \forall z R(f(a,x), y, g(z)),$$

en Prolog se representaría como

$$r(f(a,X), Y, g(Z)).$$

Observe que en la notación de Prolog, hemos omitido los cuantificadores universales. Además los símbolos de variable x, y, z de la Lógica de predicados, en Prolog los hemos escrito como X, Y, Z. Los símbolos de función f, g los hemos dejado igual, y el símbolo de predicado R lo hemos escrito como r.

Una expresión Prolog del tipo anterior se denomina un hecho. Otros ejemplos de hechos son:

```
padre(antonio,eva).
es_igual(X,cubo(raiz_cubica(X))).
es_par(8).
conoce(pepe,X).
```

El primero nos diría que antonio es el padre de eva. Aquí el contexto nos dice que padre es un símbolo de predicado.

El segundo ejemplo expresa que para todo X, X es igual al cubo de la raíz cúbica de X. Aquí, es_igual es un símbolo de predicado, mientras que cubo y raiz_cubica son símbolos de función.

El tercero expresa que 8 es un número (entero) par.

El cuarto ejemplo nos diría que la persona representada por la constante pepe conoce a todo el mundo, es decir, para todo X, pepe conoce a X. Obviamente, conoce es un símbolo de predicado ya que expresa relaciones entre objetos.

Las conectivas de la Lógica de predicados se representan en Prolog de la forma siguiente:

1. La conjunción se escribe poniendo una coma (,) entre las fórmulas atómicas, y la disyunción poniendo un punto y coma (;).

2. La negación lógica se escribe utilizando not o bien \+. Por ejemplo not(p(X)) o bien \+(p(X)) dicen que para todo X, no es cierto p(X).

3. La implicación o condicional se representa escribiendo el símbolo de dos puntos seguido por el símbolo de la resta (:-). Por ejemplo, la fórmula de la Lógica de predicados,

   $$\forall x(P(x) \rightarrow Q(x)),$$

   se escribiría como

   ```
   q(X) :- p(X).
   ```

   Nótese que en Prolog, es como si hubiésemos hecho una imagen especular de la fórmula.

Una fórmula de la Lógica de predicados del tipo

$$\forall x_1 \forall x_2 \cdots \forall x_n \Big(\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_k \rightarrow \beta \Big),$$

se traduce como

$$\beta :- \alpha_1, \alpha_2, \ldots, \alpha_k.$$

y se denomina regla.

Observe que tanto los hechos como las reglas han de terminar en un punto. Algunos ejemplos de reglas son:

```
es_pato(P) :- tiene_plumas(P), hace_cuac(P).
p :- q1, q2, q3.
cuadrilatero(X) :- poligono(X), numero_lados(X,4).
novios(X,Y) :- le_gusta_a(X,Y), le_gusta_a(Y,X).
hermana_de(X,Y) :- mujer(X),
                   es_padre_de(P,X),
                   es_madre_de(M,X),
                   es_padre_de(P,Y),
                   es_madre_de(M,Y).
```

El primer ejemplo nos dice que si un objeto designado por P tiene plumas y emite el sonido "cuac", entonces ha de ser un pato.

En el segundo ejemplo, tenemos predicados de aridad cero, y no intervienen símbolos de variable. Así, se trata de una fórmula de la Lógica de proposiciones. Nos dice que si se verifican simultáneamente q1, q2, q3, entonces también se verifica p.

El ejemplo tercero nos dice que un polígono de cuatro lados es un cuadrilátero.

El ejemplo cuarto expresa que si a X le gusta Y, y a Y le gusta X, entonces X e Y son novios.

En el ejemplo quinto, estamos diciendo que si X es mujer, y además existen P y M tales que P es padre de X, M es madre de X, P es padre de Y, y M es madre de Y, entonces X es hermana de Y.

Volviendo a las definiciones, las reglas se suelen indicar como

```
cabeza_de_la_regla :- cuerpo_de_la_regla.
```

La cabeza describe el hecho que se intenta definir; el cuerpo describe los objetivos que deben satisfacerse para que la cabeza sea cierta. De manera un poco más concreta, la regla

$$\mathsf{c} :- \mathsf{o}_1, \mathsf{o}_2, \ldots, \mathsf{o}_n.$$

nos dice que la demostración de c se sigue ó se deduce de la demostración de los objetivos $\mathsf{o}_1, \mathsf{o}_2, \ldots, \mathsf{o}_n$, es decir, para llevar a cabo una tarea c, tenemos que realizar las subtareas $\mathsf{o}_1, \mathsf{o}_2, \ldots, \mathsf{o}_n$.

Mencionamos por último que en Prolog no hay declaraciones de tipo. El significado de cada símbolo, se infiere del uso que se haga del mismo.

3.3. Programas lógicos y objetivos

Un programa en Prolog consiste en un conjunto de reglas y hechos cada uno terminado en un punto, que almacenaremos en un archivo de texto cuya extensión será `.pl`. Un programa de Prolog también se denomina **Base de datos**.

En el Escritorio de Windows tenemos dos iconos de Prolog. Uno, donde se encuentra únicamente el intérprete de Prolog, y otro que incorpora además un editor de programas de Prolog. Nosotros utilizaremos el segundo, denominado SWI-Prolog-Editor. De hecho, al hacer doble click en cualquier archivo con extensión .pl se carga con este último.

Hacemos doble click en el archivo adjunto .pl, y se abre Prolog. Vemos que la pantalla aparece dividida en dos ventanas. La superior es la ventana del editor de textos donde se muestra el archivo que hemos cargado. La inferior es la ventana del intérprete de Prolog a través de la cual introducimos los comandos ó preguntas a Prolog acerca de la Base de datos que tenemos en la ventana del editor. El indicador de Prolog para indicar que está a la espera es ?-.

Observamos que en la ventana del editor aparecen reglas y hechos. Además hay líneas con comentarios que vienen precedidas por el símbolo %.

El editor representa los símbolos de constante ó de función, de color verde, los símbolos de variable, de color azul, y los símbolos de predicado, de color negro. Un mismo símbolo puede usarse en Prolog como símbolo de predicado ó de función. Nótese que el editor asigna un color a cada uno de ellos según su significado. Además ello no supone ningún problema para Prolog.

Las preguntas u objetivos son las herramientas que tenemos para recuperar la información de la Base de datos desde Prolog. Al contrario que los hechos y reglas, los objetivos que le planteamos a Prolog se corresponden con sentencias de la Lógica de predicados cuantificadas existencialmente, como por ejemplo,

$$\exists x \exists y R(x, f(y)).$$

No introduzca la siguiente línea en Prolog, es únicamente a título informativo. Para este ejemplo, escribiríamos el objetivo

?- r(X,f(Y)).

Al hacer una pregunta a un programa lógico estamos determinando si una fórmula es consecuencia lógica del programa. Prolog considera que todo lo que hay en la Base de Datos es verdad, de manera que si Prolog responde "Yes" significa que ha podido demostrarlo, y si responde "No" es que no ha podido demostrarlo. La respuesta "No" no debe interpretarse como que la pregunta planteada es falsa, sino que con lo que Prolog conoce no puede demostrar su veracidad.

Si planteamos un objetivo sin variables, realmente le estamos preguntando si éste se encuentra o no en la Base de datos, por lo que su respuesta será "Yes" o "No", según el caso, y concluirá.

Si la pregunta que le planteamos contiene variables, el sistema trata de encontrar una solución particular, es decir, un valor particular para cada una de las variables que aparecen en la pregunta. Si tiene éxito, muestra en pantalla dichos valores y marca el lugar donde los encontró. Entonces Prolog queda a la espera de nuevas instrucciones.

- Si pulsamos " ; ", trata de encontrar la siguiente solución, mostrándola si la encuentra y otra vez quedando en espera, o bien devolviendo la respuesta "No" y acabando.
- Si pulsamos ENTER, le estamos diciendo que concluya la búsqueda.

El hecho de tener cargada la base de datos en el editor, no significa que Prolog la haya procesado. De hecho tal archivo aún no es reconocido por Prolog. Para ello hay que pulsar en el tercer botón del menú superior [Iniciar]>[Consultar].

Si no hay errores sintácticos en el archivo, Prolog da un mensaje en la ventana de comandos indicando que la carga de la base de datos ha sido satisfactoria.

Pregunte a Prolog lo siguiente:

```
?- listing(padre).
```

Le hemos preguntado por los hechos y reglas en la base de datos cuya parte izquierda es padre.

Podemos preguntarle si david es el padre de emilio.

```
?- padre(david,emilio).
```

Hemos escrito estos nombres propios en minúsculas pues así es como vienen definidos en nuestro programa. Si escribe lo siguiente, verá que el sistema no responde tal y como usted esperaba, aunque da una respuesta:

```
?- padre(David,Emilio).
```

¿Cuál es la explicación de lo que ha pasado?

Pues que David y Emilio ahora son nombres de variable y trata de encontrar un primer valor para la variable David y la variable Emilio, de forma que el primero sea el padre del segundo. Como puede apreciar, obtiene

```
David=antonio
Emilio=carlos
```

Vuelva a consultar el archivo en la ventana del editor para recordar que antonio es padre de carlos. ¿Lo entiende ahora mejor?

Puede concluir la búsqueda pulsando ENTER o bien pedirle más soluciones pulsando punto y coma.

Si escribimos

```
?- not(padre(antonio,david)).
```

le estamos preguntando si antonio no es padre de david, ante lo cual responde afirmativamente. De forma equivalente podemos escribir

```
?- \+ padre(antonio,david).
```

Ahora formulamos la siguiente pregunta: ¿quiénes son los hijos de eva? En la Lógica de predicados tendríamos la formula $\exists x M(eva, x)$ que en Prolog se traduce como:

```
?- madre(eva,X).
```

El sistema sólo encuentra

```
X=emilio
```

y acaba.

Pregúntele quién es el padre de silvia.

Podemos hacer "preguntar múltiples" en una misma línea. Recuerde el significado del punto y coma, y de la coma en Prolog. Introducimos

```
?- padre(X,carlos), madre(Y,carlos).
```

cuya respuesta es:

```
X = antonio,
Y = maria
```

El sistema queda en espera. Si introcimos punto y coma, el sistema devuelve false, indicándonos con ello que ya no ha encontrado más soluciones. Ahora introducimos

```
?- padre(X,carlos); madre(Y,carlos).
```

cuya respuesta es:

```
X = antonio
```

y el sistema queda a la espera. Si introducimos punto y coma, encuentra además

```
Y = maria
```

y queda de nuevo a la espera. Al introducir punto y coma, devuelve false.

Puesto que la cosa se va a ir complicando un poquito, sería bueno que hiciera un grafo dirigido a partir de la información suministrada por los hechos. Los nodos son las personas y trazamos un lado dirigido de la persona X a la persona Y, si X es padre o bien es madre de Y. Ahora trate de responder a las siguientes preguntas y constate las respuestas en el grafo que ha dibujado.

¿Son carlos y eva hermanos?

¿Quiénes son los hermanos de fernando?

Observe que el predicado hermanos es simétrico, es decir, el orden de los argumentos esencialmente no influye.

Vea el código del predicado hermanos y trate de entender cómo funciona y por qué es correcto.

¿Quiénes son hermanos?

¿Quiénes son los progenitores de emilio?

Utilice el predicado progenitor para preguntarle por los hijos de carlos.

A continuación vea el código del predicado abuelo.

¿Qué información podemos obtener sobre los abuelos de silvia?

¿Quiénes son los nietos de antonio?

¿Qué personas son hijos o nietos de antonio? Hágalo ejecutando un único comando.

Escriba reglas Prolog (pero no hechos) que definan las siguientes relaciones de parentesco:

`abuela(A,N)` indica si A es abuela de N;

`tioa(A,B)` indica si A es tío o tía de B;

`primos(A,B)` indica si A y B son primos.

Cree un archivo llamado parentescos.pl que incluya las reglas y hechos del archivo principios.pl así como las tres anteriores que acaba de escribir. Recuerde que hay que darle a la opción Consultar para que Prolog reconozca los cambios.

Si todavía no ha tenido bastante, escriba un archivo llamado misfamiliares.pl similar al archivo parentescos.pl pero ahora escribiendo como hechos la información sobre sus familares. Puede poner tantos niveles de ascendientes o de descendientes como quiera. A continuación, pregúntele a Prolog.