Paradigma Lógico

Módulo 1:
Predicados. Individuos.
Consultas. Universo Cerrado.

por Fernando Dodino
Carlos Lombardi
Nicolás Passerini
Daniel Solmirano

revisado por Matías Freyre

Versión 2.1
Junio 2019

Contenido

1 Introducción al paradigma lógico

Comenzaremos a estudiar un nuevo paradigma, el paradigma lógico. Recordemos que los paradigmas nos dan un marco conceptual para resolver problemas; en el caso de la programación, nos dicen cómo modelar las estructuras de datos y los algoritmos.

Hemos visto anteriormente dos visiones distintas para construir software:

- En la visión **imperativa o procedimental** yo defino una secuencia de pasos a ejecutar y guardo estados intermedios en posiciones de memoria que llamo variables. Esto respeta fielmente el modelo matemático de Von Neumann.
- En la visión **declarativa** no hay algoritmo. Sólo definiciones (el qué) y hay alguna magia detrás de todo esto que lo termina resolviendo. Si esa magia es transparente => me puedo concentrar en lo que es esencial al problema (ése es el beneficio de la abstracción).

2 Definiendo nuestro conocimiento

Para mostrar cómo son las reglas de juego del paradigma lógico, comenzaremos con un ejemplo sencillo que modela comidas: en un primer paso queremos decir que los ravioles y los fideos son pastas. Esto lo escribimos en una **base de conocimiento**, que define el alcance, lo que forma parte de nuestro universo de cosas conocidas y lo que no.

```
pastas(ravioles).
pastas(fideos).
```

comidas.pl --> nuestra base de conocimientos

3 Predicados e individuos

La base de conocimiento se conforma de predicados: el predicado *pastas* tiene aridad 1, esto significa que un solo individuo participa de la relación, por eso se escribe **pastas/1**. A los predicados que tienen un solo argumento se los llama monádicos, y expresan características o atributos de los individuos en cuestión.

Los individuos son los elementos que forman parte del universo posible de los predicados. En el ejemplo anterior, tanto ravioles como fideos son individuos posibles que satisfacen la relación pastas/1. Es natural asociar la idea de individuo a dato, más adelante profundizaremos en este aspecto.

4 Consultas

La **lógica** es una ciencia formal que estudia los principios de la demostración e inferencia válida. La inferencia es el proceso por el cual se derivan conclusiones a partir de premisas. Cuando ejecutamos un programa lógico nos aparece una consola para disparar consultas que toman como entrada las definiciones de la base de conocimiento para extraer conclusiones.

```
? pastas(ravioles).
true

? pastas(fideos).
true
```

Es importante el punto para delimitar el final de una consulta, de lo contrario en la consola Prolog nos aparecerá un símbolo pipe (|) que indicará que podemos seguir escribiendo la consulta en más de una línea:

```
? pastas(ravioles)
| .
true
```

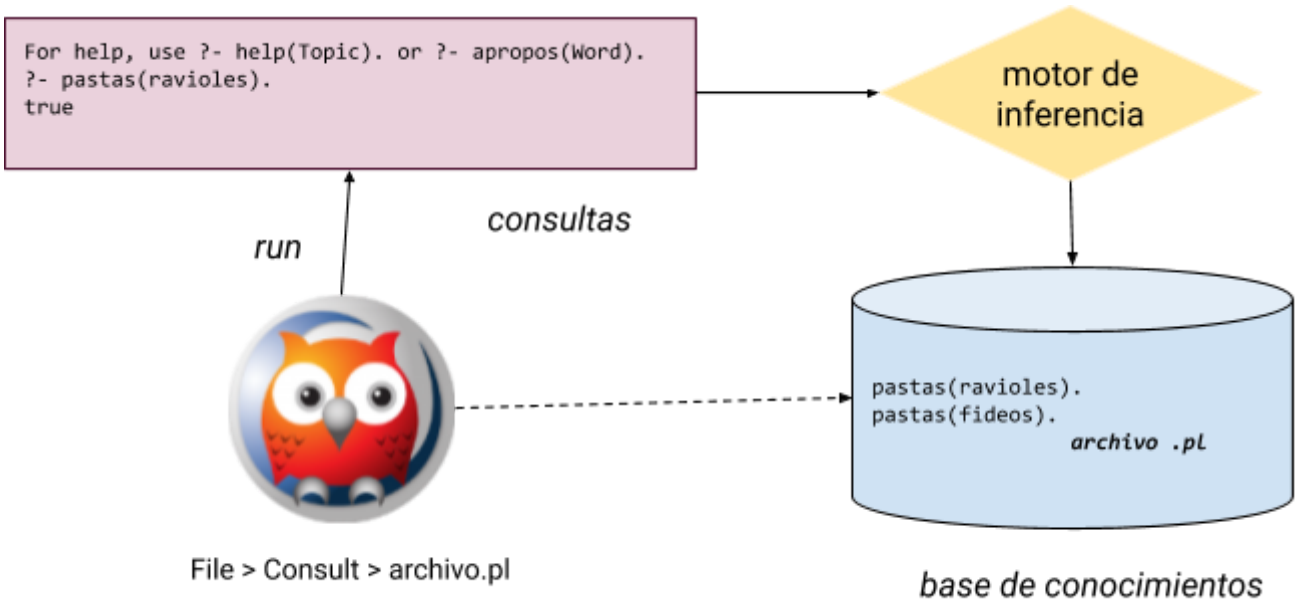
Al escribir un punto en la segunda línea, Prolog evalúa la consulta considerando como input todas las líneas escritas. Es equivalente a hacer la consulta `pastas(ravioles).` incluyendo el punto, claro está.

5 Una aplicación en el paradigma lógico

Una aplicación en el Paradigma Lógico es una solución **declarativa** porque

- declaro conocimiento a través de los predicados, en un archivo *.pl*, utilizando cualquier editor de texto[1]
- en las consultas, un motor de inferencia (en este caso de Prolog) permite sacar conclusiones a partir de ese conocimiento. El motor es el que termina construyendo el algoritmo, para poder resolver las consultas.

[1] Para más información vean en el sitio web http://pdep.com.ar/Home/software/editores



6 Principio de Universo Cerrado

Si hacemos las siguientes consultas

```
? pastas(pechitoCerdo).
false

? pastas(ñoquis).
false
```

vemos que Prolog nos dice que ni el pechito de cerdo ni los ñoquis son pastas. Esto puede verse de dos maneras:

- los ñoquis no son pastas: la afirmación “los ñoquis son pastas” es **falsa**
- no puedo probar que los ñoquis sean pastas: la afirmación “los ñoquis son pastas” es **desconocida**.

Al comenzar a escribir nuestra base de conocimientos contamos que estábamos definiendo nuestro alcance, el universo de los elementos conocidos, por lo tanto hay dos opciones:

- considerar tres estados posibles: cierto, falso y desconocido: estos son los lenguajes que trabajan con el **Principio de Universo Abierto**
- considerar a todo lo desconocido como falso, trabajar con el **Principio de Universo Cerrado**: los lenguajes como Prolog entran en esta categoría

De esa manera, todo lo que no podamos probar será considerado falso. No tendrá sentido por cierto escribir que el asado no es una pasta, que no me fui de vacaciones a

Claromecó, que no tengo 23 años... todos estos hechos pueden asumirse falsos por el solo hecho de no estar en la base de conocimientos.

A partir de acá, vamos a llamar *hecho* a una afirmación codificada en la base de conocimiento, con la sintaxis que vimos antes, y que por lo tanto asumimos como verdadera. Cada hecho es una cláusula. Más adelante vamos a ampliar esta definición.

Ejercicio: definir que
1. los involtinis son pastas
2. las berenjenas en escabeche no son pastas

Solución:
```
pastas(involtinis).
// no tiene sentido escribir lo que no es
// si las berenjenas en escabeche no son pastas, no escribimos nada
```

7 Predicados poliádicos

Hemos visto anteriormente que los predicados monádicos son los que tienen aridad 1. Los predicados que tienen más de un argumento se llaman poliádicos, porque expresan relaciones entre individuos.

```
come(juan, ravioles).
come(brenda, fideos).
gusta(brenda, fideos).
```

Tip: es conveniente escribir las definiciones de un mismo predicado en forma contigua, y no intercalar definiciones de otros predicados. Si la base de conocimientos se escribe así:

```
come(juan, ravioles).
gusta(brenda, fideos).    ← el predicado gusta/2 está en el medio de come/2
come(brenda, fideos).
```

El compilador Prolog emite un mensaje de advertencia:

```
Warning: /home/fernando/workspace/prolog-2017/comidas.pl:3:
  Clauses of come/2 are not together in the source-file
 Earlier definition at /home/fernando/workspace/prolog-2017/comidas.pl:1
  Current predicate: gusta/2
  Use :- discontiguous come/2. to suppress this message
```

Si bien podemos solucionarlo avisando al compilador que las *cláusulas* que definen este *predicado* no están escritos en forma contigua, vamos a preferir ser ordenados de aquí en más.

7.1 Consultas sobre predicados poliádicos

Volvemos a nuestra anterior definición

```
come(juan, ravioles).
come(brenda, fideos).
gusta(brenda, fideos).
```

Realicemos algunas consultas sobre la base de conocimientos:

```
? come(brenda, fideos).
true

? come(fideos, brenda).
false

? gusta(brenda, fideos).
true

? gusta(brenda, juan).
false

? come(brenda, fideos, ricos).
ERROR: Undefined procedure: come/3
ERROR:     However, there are definitions for:
ERROR:         come/2
false.
```

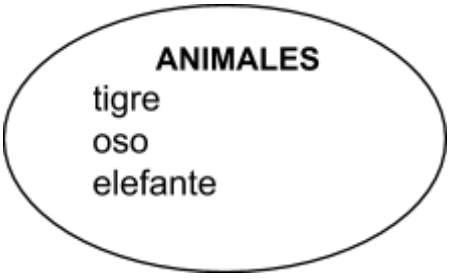
Aquí vemos que

- las consultas deben respetar la cantidad de argumentos con la que se definió el predicado. De lo contrario veremos un mensaje de error.
- el orden de los argumentos es importante: `come(brenda, fideos)`se satisface mientras que si consultamos `come(fideos, brenda)` no será cierto.
    - Las relaciones no son bidireccionales por defecto, esto debemos tenerlo en cuenta a la hora de hacer nuestras definiciones.
    - El sentido de los argumentos se los damos nosotros, también hay que estar atento al lugar que ocupa cada uno de los individuos involucrados en un predicado.

8 Definición por extensión y por comprensión

Un conjunto de hechos para el mismo predicado forman la definición por extensión del predicado.

Ejemplo:




```
animal(tigre).
animal(oso).
animal(elefante).
```

Es como decir *Animales = { tigre, oso, elefante }*

```
planeta(venus).
planeta(tierra).
```

¿Cuál es la desventaja de este tipo de definición? Requiere la enumeración de todos los elementos que componen el conjunto. Para conjuntos con muchos elementos es, cuanto menos, incómodo. Otra desventaja que presenta es que no brinda información referente al conjunto más allá de la que el observador le puede dar.

Más adelante veremos cómo definir un predicado por comprensión.

9 ¿Dónde aplicamos la lógica en computación?

- Bases de datos: lenguajes de restricciones, lenguajes de consulta (SQL)
- Inteligencia artificial: representación del conocimiento, deducción.
- Ingeniería de software: especificación de sistemas (lenguajes Z)
- Criptografía: verificación de protocolos criptográficos
- Procesamiento del lenguaje natural

10 Resumen del capítulo

En nuestra primera aproximación al Paradigma Lógico, hemos visto que las soluciones que vamos a encarar son declarativas: escribimos conocimiento partiendo de las características de los individuos o de las relaciones entre ellos en una base de conocimientos. Luego podemos realizar diferentes consultas delegando al motor de inferencia la forma en que finalmente lo resuelve.

La base de conocimientos compone todo el universo conocido, todo lo que está fuera no se puede probar que existe, por lo tanto se asume falso según el Principio de Universo Cerrado.