Programación lógica

Pedro O. Pérez M., PhD.

Implementación de métodos computacionales
Tecnológico de Monterrey

pperezm@tec.mx

02-2023

- La programación que utiliza una forma de lógica simbólica como lenguaje de programación a menudo se denomina programación lógica, y los lenguajes basados en lógica simbólica se denominan lenguajes de programación lógica o lenguajes declarativos.
- Esto significa que solo se establecen las especificaciones de los resultados deseados en lugar de los procedimientos detallados para producirlos.
- Los programas en los lenguajes de programación lógica son colecciones de hechos y reglas.
- La sintaxis de los lenguajes de programación lógica es notablemente diferente de la de los lenguajes imperativos y funcionales.

- Se puede pensar en una **proposición** como una declaración lógica que puede o no ser verdadera. Se compone de objetos y las relaciones entre los objetos. La lógica formal se desarrolló para proporcionar un método para describir proposiciones, con el objetivo de permitir que se verificara la validez de esas proposiciones formuladas formalmente.
- La **lógica simbólica** se puede utilizar para las tres necesidades básicas de la lógica formal: expresar proposiciones, expresar las relaciones entre proposiciones y describir cómo se pueden inferir nuevas proposiciones a partir de otras proposiciones que se supone que son verdaderas.
- La forma particular de lógica simbólica que se utiliza para la programación lógica se denomina **cálculo de predicados de primer orden**.

- Una constante es un símbolo que representa un objeto. Una variable es un símbolo que puede representar diferentes objetos en diferentes momentos, aunque en un sentido mucho más cercano a las matemáticas que a las variables en un lenguaje de programación imperativo.
- Las proposiciones más simples, que se llaman **proposiciones atómicas**, consisten en **términos compuestos**. Un término compuesto es un elemento de una relación matemática, escrito en una forma que tiene la apariencia de notación de función matemática.

- Un término compuesto se compone de dos partes: un **funtor**, que es el símbolo de función que nombra la relación, y una **lista ordenada de parámetros**, que juntos representan un elemento de la relación.

```
man(jake).
man(fred).
like(jake, meat).
```

- Las proposiciones se pueden enunciar de dos modos: uno en el que la proposición se define como verdadera y otro en el que la verdad de la proposición es algo que debe determinarse. En otras palabras, se puede afirmar que las proposiciones son hechos o consultas. Las proposiciones de ejemplo podrían ser cualquiera.

Name	*Symbol*	*Example*	*Meaning*
negation	$\neg$	$\neg$ a	not *a*
conjunction	$\cap$	$a \cap b$	*a* and *b*
disjunction	$\cup$	$a \cup b$	*a* or *b*
equivalence	$\equiv$	$a \equiv b$	*a* is equivalent to *b*
implication	$\supset$	$a \supset b$	*a* implies *b*
	$\subset$	$a \subset b$	*b* implies *a*

Name	*Example*	*Meaning*
universal	$\forall X.P$	For all X, P is true.
existential	$\exists X.P$	There exists a value of X such that P is true.

- La **resolución** es una regla de inferencia que permite calcular proposiciones inferidas a partir de proposiciones dadas, proporcionando así un método con aplicación potencial para la demostración automática de teoremas. La resolución fue ideada para ser aplicada a las proposiciones en forma de cláusula. El concepto de resolución es el siguiente: Supongamos que hay dos proposiciones con las formas.

$$P_1 \subset P_2$$
$$Q_1 \subset Q_2$$

$$T \subset P_2$$
$$Q_1 \subset T$$

- La resolución es en realidad más compleja de lo que ilustran estos simples ejemplos. En particular, la presencia de variables en las proposiciones requiere resolución para encontrar valores para esas variables que permitan que el proceso de emparejamiento tenga éxito. Este proceso de determinar valores útiles para las variables se denomina **unificación**. La asignación temporal de valores a las variables para permitir la unificación se denomina instanciación.
- Es común que el proceso de resolución cree instancias de una variable con un valor, no complete la coincidencia requerida y luego deba retroceder e instanciar la variable con un valor diferente.
- Una propiedad críticamente importante de la resolución es su capacidad para detectar cualquier inconsistencia en un conjunto dado de proposiciones. Esto se basa en la propiedad formal de la resolución denominada **refutación completa**. Lo que esto significa es que dado un conjunto de proposiciones inconsistentes, la resolución puede demostrar que lo son.

- Una forma de simplificar el proceso de resolución es restringir la forma de las proposiciones. Una restricción útil es exigir que las proposiciones sean **cláusulas de Horn**. Las cláusulas de cuerno pueden tener solo dos formas: tienen una sola proposición atómica en el lado izquierdo o un lado izquierdo vacío. El lado izquierdo de una proposición de forma clausal a veces se denomina cabeza, y las cláusulas Horn con lados izquierdos se denominan cláusulas Horn con cabeza.

```
man(jake).
man(fred).
like(jake, meat).
```

Descripción general de la programación lógica

- Los lenguajes utilizados para la programación lógica se denominan lenguajes declarativos, porque los programas escritos en ellos consisten en declaraciones en lugar de asignaciones y declaraciones de flujo de control. Estas declaraciones son en realidad enunciados o proposiciones en lógica simbólica.

- Una de las características esenciales de los lenguajes de programación lógica es su semántica, que se denomina **semántica declarativa**. El concepto básico de esta semántica es que hay una manera simple de determinar el significado de cada declaración, y no depende de cómo se pueda usar la declaración para resolver un problema. Por ejemplo, el significado de una proposición dada en un lenguaje de programación lógica puede determinarse de manera concisa a partir de la declaración misma. En un lenguaje imperativo, la semántica de una declaración de asignación simple requiere el examen de las declaraciones locales, el conocimiento de las reglas de alcance del lenguaje y posiblemente incluso el examen de programas en otros archivos solo para determinar los tipos de variables en la declaración de asignación. Luego, suponiendo que la expresión de la asignación contiene variables, se debe rastrear la ejecución del programa antes de la declaración de asignación para determinar los valores de esas variables. La acción resultante de la declaración, entonces, depende de su contexto de tiempo de ejecución. Comparando esta semántica con la de una proposición en un lenguaje lógico, sin necesidad de considerar el contexto textual o las secuencias de ejecución, queda claro que la semántica declarativa es mucho más simple que la semántica de los lenguajes imperativos.

- La programación tanto en lenguaje imperativo como funcional es principalmente procedimental, lo que significa que el programador sabe lo que debe lograr un programa e instruye a la computadora sobre cómo se debe realizar el cálculo exactamente.
- La programación en un lenguaje de programación lógica no es procedimental. Los programas en tales lenguajes no indican exactamente cómo se debe calcular un resultado, sino que describen la forma del resultado. La diferencia es que suponemos que el sistema informático puede determinar de algún modo cómo se calculará el resultado.

- Alain Colmerauer y Phillippe Roussel de la Universidad de Aix-Marseille, con la ayuda de Robert Kowalski de la Universidad de Edimburgo, desarrollaron el diseño fundamental de Prolog. Colmerauer y Roussel estaban interesados en el procesamiento del lenguaje natural y Kowalski estaba interesado en la demostración automática de teoremas. La colaboración entre la Universidad de Aix-Marseille y la Universidad de Edimburgo continuó hasta mediados de la década de 1970. Desde entonces, la investigación sobre el desarrollo y el uso del lenguaje ha progresado de forma independiente en esos dos lugares, dando como resultado, entre otras cosas, dos dialectos sintácticamente diferentes de Prolog.

- Un **término** de Prolog es una constante, una variable o una estructura.
- Una **constante** es un átomo o un número entero. Los átomos son los valores simbólicos de Prolog y son similares a sus contrapartes en Scheme. En particular, un átomo es una cadena de letras, dígitos y guiones bajos que comienza con una letra minúscula o una cadena de caracteres ASCII imprimibles delimitados por apóstrofes.

- Una **variable** es cualquier cadena de letras, dígitos y guiones bajos que comienza con una letra mayúscula o un guión bajo (_). Las variables no están vinculadas a tipos por declaraciones. La vinculación de un valor, y por tanto de un tipo, a una variable se denomina **instanciación**. La instanciación ocurre solo en el proceso de resolución. Una variable a la que no se le ha asignado un valor se denomina **no instanciada**. Las instanciaciones duran solo lo que se necesita para satisfacer un objetivo completo, que implica la prueba o refutación de una proposición. Las variables de Prolog son solo parientes lejanos, en términos tanto de semántica como de uso, de las variables en los lenguajes imperativos.

- El último tipo de término se llama estructura. Las estructuras representan las proposiciones atómicas del cálculo de predicados, y su forma general es la misma:

```
funtor(lista de parámetros).
```

- El funtor es cualquier átomo y se utiliza para identificar la estructura. La lista de parámetros puede ser cualquier lista de átomos, variables u otras estructuras.
- También se pueden considerar como objetos, en cuyo caso permiten establecer hechos en términos de varios átomos relacionados. En este sentido, las estructuras son relaciones, pues establecen relaciones entre términos. Una estructura también es un predicado cuando su contexto especifica que es una consulta (pregunta).