Lenguaje de programación lógico e interpretado (Prolog). La programación lógica es un paradigma de los lenguajes de programación en el cual los programas se consideran como una serie de aserciones lógicas. De esta forma, el conocimiento se representa mediante reglas, tratándose de sistemas declarativos.

Contenido

[ocultar]

Historia

Prolog es un lenguaje de programación simple pero poderoso desarrollado en la Universidad de Aix-Marseille (Marsella, Francia) por los profesores Alain Colmerauer y Philippe Roussel, como una herramienta práctica para programación lógica. Nació de un proyecto que no tenía como objetivo la implementación de un lenguaje de programación, sino el procesamiento de lenguajes naturales. Alain Colmerauer y Robert Pasero trabajaban en la parte del procesado del lenguaje natural y Jean Trudel y Philippe Roussel en la parte de deducción e inferencia del sistema. Interesado por el método de resolución SL, Trudel persuadió a Robert Kowalski para que se uniera al proyecto, dando lugar a una versión preliminar del lenguaje Prolog a finales de 1971 y apareciendo la versión definitiva en 1972. Esta primera versión de Prolog fue programada en ALGOL W.Inicialmente se trataba de un lenguaje totalmente interpretado hasta que, en 1983, David H.D. Warren desarrolló un compilador capaz de traducir Prolog en un conjunto de instrucciones de una máquina abstracta denominada Warren Abstract Machine, o abreviadamente, WAM. Desde entonces Prolog es un lenguaje semi-interpretado.

Si bien en un principio se trataba de un lenguaje de uso reducido, la aparición de intérpretes del mismo para microordenadores de 8 bits (ej: micro-PROLOG) y para ordenadores domésticos de 16 bits (ej: Turbo PROLOG de Borland, entre otros muchos) a lo largo de la década de 1980 contribuyó notablemente a su popularización.Otro importante factor en su difusión fue la adopción del mismo para el desarrollo del proyecto de la quinta generación de computadoras a principios de

la década de los 1980, en cuyo contexto se desarrolló la implementación paralelizada del lenguaje llamada KL1 y del que deriva parte del desarrollo moderno de Prolog. Las primeras versiones del lenguaje diferían, en sus diferentes implementaciones, en muchos aspectos de sus sintaxis, empleándose mayormente como forma normalizada el dialecto propuesto por la Universidad de Edimburgo , hasta que en 1995 se estableció un estándar ISO (ISO/IEC 13211-1), llamado ISO-Prolog.

Entorno de desarrollo Prolog

Prolog es un lenguaje de programación seminterpretado. Su funcionamiento es muy similar a Java. El código fuente se compila a un código de byte el cuál se interpreta en una máquina virtual denominada Warren Abstract Machine (comúnmente denominada WAM).

Por eso, un entorno de desarrollo Prolog se compone de:

- **Un compilador:** Transforma el código fuente en código de byte. A diferencia de Java, no existe un Standard al respecto. Por eso, el código de byte generado por un entorno de desarrollo no tiene por que funcionar en el intérprete de otro entorno.
- **Un intérprete**: Ejecuta el código de byte. Un shell o top-level. Se trata de una utilidad que permite probar los programas, depurarlos, etc. Su funcionamiento es similar a los interfaces de línea de comando de los sistemas operativos.
- **Una biblioteca de utilidades:** Estas bibliotecas son, en general, muy amplias. Muchos entornos incluyen (afortunadamente) unas bibliotecas standard-ISO que permiten funcionalidades básicas como manipular cadenas, entrada/salida, etc.

Generalmente, los entornos de desarrollo ofrecen extensiones al lenguaje como pueden ser la programación con restricciones, concurrente, orientada a objetos, etc. Sería injusto no mencionar aquí el entorno de desarrollo más popular: SICStus Prolog, si bien, se trata de un entorno de desarrollo comercial (no gratuito).

SICStus, CIAO Prolog, y posiblemente otros más, ofrecen entornos integrados generalmente basados en Emacs que resultan muy fáciles de usar. CIAO Prolog además ofrece un auto documentador similar al existente para Java además de un preprocesador de programas. Prácticamente todos ellos son multiplataforma.

Elementos en Prolog

Como hemos especificado antes, para construir programas en Prolog necesitamos una serie de elementos. Vamos a especificarlos:

- **Átomos:** Es una definición genérica de un objeto del mundo que queremos representar.

- **Predicados:** Nos permite especificar características de los objetos de nuestro mundo o las relaciones entre ellos.

Hechos

Es algo que está ocurriendo en el mundo, característica o relación entre objetos. En el lenguaje natural un hecho podría ser por ejemplo que Lógica y Compatibilidad es una asignatura de Ingeniería Informática. Expresan relaciones entre objetos. Suponiendo que se quiera expresar el hecho de que "un coche tiene ruedas". Este hecho, consta de dos objetos, "coche" y "ruedas", y de una relación llamada "tiene". La forma de representarlo en PROLOG es: tiene(coche,ruedas).

- Los nombres de objetos y relaciones deben comenzar con una letra minúscula.
- Primero se escribe la relación, y luego los objetos separados por comas y encerrados entre paréntesis.
- Al final de un hecho debe ir un punto (".").

El orden de los objetos dentro de la relación es arbitrario, pero debemos ser coherentes a lo largo de la base de hechos.

Variables

No es variable con el concepto que se tiene de ella en la programación habitual. En Prolog, una variable representa el valor de un Átomo. Representan objetos que el mismo PROLOG determina. Una variable puede estar instanciada ó no instanciada. Estar instanciada cuando existe un objeto determinado representado por la variable. Los nombres de variables comienzan siempre por una letra mayúscula.

Un caso particular es la variable anónima, representada por el carácter subrayado ("_"). Es una especie de comodín que utilizaremos en aquellos lugares que debería aparecer una variable, pero no nos interesa darle un nombre concreto ya que no vamos a utilizarla posteriormente.

Reglas

Las reglas se utilizan en PROLOG para significar que un hecho depende de uno ó mas hechos. Son la representación de las implicaciones lógicas del tipo p ---> q (p implica q).

- Una regla consiste en una cabeza y un cuerpo, unidos por el signo ":-".
- La cabeza está formada por un único hecho.
- El cuerpo puede ser uno ó mas hechos (conjunción de hechos), separados por una coma (","), que actúa como el "y" lógico.
- Las reglas finalizan con un punto (".").

La cabeza en una regla PROLOG corresponde al consecuente de una implicación lógica, y el cuerpo al antecedente. Este hecho puede conducir a errores de

representación. Supongamos el siguiente razonamiento lógico: **tiempo(lluvioso) ---> suelo(mojado) suelo(mojado)**

Que el suelo está, mojado, es una condición suficiente de que el tiempo sea lluvioso, pero no necesaria. Por lo tanto, a partir de ese hecho, no podemos deducir mediante la implicación, que está, lloviendo (pueden haber regado las calles). La representación correcta en PROLOG, sería:

suelo(mojado):- tiempo(lluvioso). suelo(mojado).

Adviértase que la regla está "al revés". Esto es así por el mecanismo de deducción hacia atrás que emplea PROLOG. Si cometiéramos el error de representarla como:

tiempo(lluvioso):- suelo(mojado). suelo(mojado).

PROLOG, partiendo del hecho de que el suelo está mojado, deduciría incorrectamente que el tiempo es lluvioso. Para generalizar una relación entre objetos mediante una regla, utilizaremos variables. Por ejemplo:

Representación lógica Representación PROLOG **es_un_coche(X) ----> tiene(X,ruedas)**

tiene(X,ruedas):- es_un_coche(X).

Con esta regla generalizamos el hecho de que cualquier objeto que sea un coche, tendrá ruedas. Al igual que antes, el hecho de que un objeto tenga ruedas, no es una condición suficiente de que sea un coche. Por lo tanto la representación inversa sería incorrecta.

Programando en Prolog

A la hora de programar en Prolog se tiene dos cuerpos principales: la especificación de los hechos y las preguntas sobre esos objetos o relaciones. Cuando se crea una base de datos con esa especificación de hechos se puede poner a hacer preguntas sobre esa especificación dando como resultado sí o no.

Prolog saca la respuesta explorando cada uno de los hechos introducidos en la base de datos hasta encontrar uno que coincida, que será el caso en la que la respuesta será afirmativa, o hasta que termine toda la base de datos, cuyo caso dará una respuesta negativa. Las preguntas que hacemos sobre la base de hechos pueden ser más complejas usando operadores lógicos como AND, OR y NOT. En este caso, Prolog busca que la satisfacción a la primera parte de la pregunta y si lo es, lo busca en la segunda.

Objetos de datos

Tipos de datos primitivos: variables y constantes:

- Enteros
- Reales

- Caracteres

Los identificadores con minúscula representan hechos, los que van con mayúscula variables. El alcance de una variable es la regla donde aparece. Tipos de datos estructurados:

- Átomos: constantes y variables de cadena.
- Listas, representadas entre [ ].
- Tipos definidos por el usuario. Las reglas para definir relaciones pueden actuar como tipos de usuario.

Ventajas y desventajas de la Programación Lógica

Ventajas

1. La habilidad de PROLOG para calcular de forma procedural es una de las ventajas específicas que tiene el lenguaje. Como consecuencia esto anima al programador a considerar el significado declarativo de los programas de forma relativamente independiente de su significado procedural. Es decir, las ventajas de la forma declarativa de este lenguaje son claras (es más fácil pensar las soluciones y muchos detalles procedurales son resueltos automáticamente por el propio lenguaje) y podemos aprovecharlas.
2. Una ventaja desde el punto de vista del usuario es la facilidad para programar ya que se pueden escribir programas rápidamente, con pocos errores originando programas claramente legibles, aun si no se conoce muy bien el lenguaje.
3. No hay que pensar demasiado en la solución del problema, ya que Prolog infiere sus respuestas basándose en las reglas declaradas dentro del programa. 4. Modularidad: cada predicado (procedimiento) puede ser ejecutado, validado y examinado independiente e individualmente. Prolog no tiene variables globales, ni asignación. Cada relación está auto contenida, lo que permite una mayor modularidad, portabilidad y reusabilidad de relaciones entre programas. 5. Polimorfismo: se trata de un lenguaje de programación sin tipos, lo que un alto nivel de abstracción e independencia de los datos (objetos).
4. En Prolog, se puede representar incluso los mismos programas Prolog como estructuras.
5. Prolog utiliza un mecanismo de búsqueda independiente de la base de hechos. Aunque pueda parecer algo retorcido, es una buena estrategia puesto que garantiza el proceso de todas las posibilidades. Es útil para el programador conocer dicho mecanismo a la hora de depurar y optimizar los programas. 8. Manejo dinámico y automático de memoria.

6. En prolog se utiliza notación prefija e infija mientras que en Lisp solo utiliza notación prefija.

Desventajas

1. La resolución automática no siempre es eficiente, por lo que eventualmente se podría dar una respuesta incorrecta a una consulta. 2. Poco eficientes. 3. Poco utilizado en aplicaciones reales.
2. Prolog algunas veces es incapaz de reconocer que un problema es (para su propio conocimiento) inaplicable o insuficiente. Si el programa no contiene suficiente información para contestar una consulta, es incapaz de reconocerlo y responde no. En esta situación sería más eficiente conocer que la respuesta no es negativa, sino que no es posible inferir un resultado.
3. Los motores de inferencia poseen algunos límites.

Compatibilidad ISO-Prolog

Existe un Standard ISO que dicta las típicas normas con respecto a la sintaxis del lenguaje y a las bibliotecas básicas que se deben ofrecer. Actualmente el Standard no contempla todos los aspectos del lenguaje, y además, no todos los entornos siguen el Standard al pie de la letra. Por eso, programas que funcionan en unos entornos podrían no funcionar en otros, o lo que es peor, funcionar de forma diferente.