Programación IV

Programación lógica.

Programación lógica

La programación funcional se basa en el concepto de función (que no es más que una evolución de los predicados), de corte más matemático. La programación lógica gira en torno al concepto de predicado, o relación entre elementos.

La lógica matemática es la manera más sencilla, para el intelecto humano, de expresar formalmente problemas complejos y de resolverlos mediante la aplicación de reglas, hipótesis y teoremas. De ahí que el concepto de "programación lógica" resulte atractivo en diversos campos donde la programación tradicional es un fracaso.

Campos de aplicación

La programación lógica encuentra su hábitat natural en aplicaciones de inteligencia artificial o relacionadas:

- Sistemas expertos, donde un sistema de información imita las recomendaciones de un experto sobre algún dominio de conocimiento.
- Demostración automática de teoremas, donde un programa genera nuevos teoremas sobre una teoría existente.
- Reconocimiento de lenguaje natural, donde un programa es capaz de comprender (con limitaciones) la información contenida en una expresión lingüística humana.

La programación lógica también se utiliza en aplicaciones más "mundanas" pero de manera muy limitada, ya que la programación tradicional es más adecuada a tareas de propósito general.

Fundamentos

La mayoría de los lenguajes de programación lógica se basan en la teoría **lógica de primer orden**, aunque también incorporan algunos comportamientos de orden superior como la lógica difusa. En este sentido, destacan los lenguajes funcionales, ya que se basan en el cálculo lambda, que es la única teoría lógica de orden superior que es demostradamente computable (hasta el momento).

Conceptos

Un concepto importante de programación lógica es la descomposición de programas en sus componentes lógicos y sus componentes de control. Con lenguajes de programación lógica de bajo nivel, estos componentes determinan la solución del problema, por eso los componentes de control pueden variar para proporcionar alternancia de ejecución de un programa lógico. Estos conceptos son capturados con el eslogan

Algoritmo= lógica + control

donde "lógica" representa un programa lógico y "control" diferentes estrategias de demostración del teorema.

PROLOG

PROLOG es un lenguaje de programación declarativo. Los lenguajes declarativos se diferencian de los lenguajes imperativos o procedurales en que están basados en formalismos abstractos (PROLOG está basado en la lógica de predicados de primer orden y LISP, otro lenguaje de programación declarativa, en lambda calculo), y por tanto su semántica no depende de la máquina en la que se ejecutan. Las sentencias en estos lenguajes se entienden sin necesidad de hacer referencia al nivel máquina para explicar los efectos colaterales.

PROLOG

PROLOG es un lenguaje de programación muy útil para resolver problemas que implican objetos y relaciones entre objetos. Está basado en los siguientes mecanismos básicos, que se irán explicando :

- Unificación
- Estructuras de datos basadas en árboles
- Backtracking automático

La sintaxis del lenguaje consiste en lo siguiente:

- Declarar hechos sobre objetos y sus relaciones
- Hacer preguntas sobre objetos y sus relaciones
- Definir reglas sobre objetos y sus relaciones

Los hechos PROLOG

Para explicar los fundamentos de PROLOG vamos a utilizar el típico ejemplo de las relaciones familiares.
Para decir que Laura es uno de los dos progenitores de Damián, podríamos declarar el siguiente hecho PROLOG:

assert(progenitor(laura, damian)).

Con assert podemos registrar un hecho. “progenitor” es el nombre de la relación o nombre de predicado y “laura” y “damian” son los argumentos. Los hechos acaban siempre con punto. Nosotros interpretaremos que Laura, primer argumento de la relación, es la madre de Damián, segundo argumento de la relación.

Las preguntas PROLOG

Sobre un conjunto de hechos se pueden realizar una serie de preguntas. Por ejemplo:

?- progenitor(laura, damian).
true.
?- progenitor(juan, damian).
false.

PROLOG busca automáticamente en la base de datos si existe un hecho que se puede unificar

Las reglas en PROLOG

Existe en PROLOG la posibilidad de definir la relación “abuelo(X,Y)” o la relación “tio(X,Y)” como reglas, además de poderlo hacer como hechos o como conjunción de objetivos.

```
[user].
abuelo(X,Y):-
progenitor(X,Z), progenitor(Z,Y).
tio(X,Y):-
progenitor(Z,Y), progenitor(V,Z), progenitor(V,X), Z /==X.


<EOF>
```

Las reglas en PROLOG

Las cláusulas PROLOG son de tres tipos: hechos, reglas y preguntas. Las cláusulas PROLOG consisten en una cabeza y un cuerpo. Los hechos son cláusulas que tienen cabeza pero no tienen cuerpo. Las preguntas sólo tienen cuerpo. Las reglas tienen siempre cabeza y cuerpo. Los hechos son siempre ciertos. Las reglas declaran cosas que son ciertas dependiendo de una condición. El programa PROLOG (o base de datos PROLOG) está formado por hechos y reglas y para PROLOG no hay ninguna distinción entre ambas. Las preguntas se le hacen al programa para determinar qué cosas son ciertas.

Las reglas en PROLOG

Tambien podemos crear reglas recursivas :

```
[user].
predecesor(X,Y):-progenitor(X,Y).
predecesor(X,Y):-progenitor(X,Z), predecesor(Z,Y).
```

La definición de varias reglas con el mismo nombre de relación equivale en PROLOG a la “O” lógica o disyunción.

Sintaxis en Prolog

Los objetos o términos PROLOG pueden ser objetos simples o estructuras. Los objetos simples pueden ser constantes o variables. Las constantes serán átomos o números. Los átomos empiezan con letra minúscula (nunca con números), pueden contener caracteres especiales y pueden ser nombres entre comillas simples. Los números serán enteros o reales, sin una definición explícita de tipos. PROLOG se utiliza para una programación simbólica, no numérica, por eso los enteros se utilizarán por ejemplo para contar el número de elementos de una lista, pero los reales son poco utilizados. Las variables empiezan con mayúscula o con subrayado. Las variables anónimas son aquellas cuyo nombre es sólo el carácter subrayado (_). Se usan cuando no es importante el nombre de la variable o cuando la variable no puede unificar con otra, dentro de la misma cláusula.

Preguntas


UNIVERSIDAD
ADVENTISTA
DEL PLATA