PROGRAMACIÓN DECLARATIVA

PROGRAMACIÓN LÓGICA

Tema PL1: Introducción al paradigma de la Programación Lógica

Grado en Ingeniería Informática

URJC

Ana Pradera

Contenido

INTRODUCCIÓN

Este tema está dedicado a describir brevemente los aspectos fundamentales del paradigma de la Programación Lógica, en particular:

- Sus fundamentos teóricos:

  Lógica Matemática. Demostración Automática.

- Su funcionamiento:

  Esquema general de funcionamiento. Ejemplos.

- Sus características y aplicaciones:

  Propiedades. Diferencias con otros paradigmas. Aplicaciones típicas.

FUNDAMENTOS TEÓRICOS

Programación Lógica

=

uso de la Lógica Matemática y de la Demostración Automática como herramientas de **cómputo**

- La Lógica Matemática se usa para la **representación del conocimiento** (enfoque *declarativo*) y para la **resolución de problemas**.
- Los sistemas de demostración de la Demostración Automática se usan para **realizar cómputos** de forma **eficiente**.

Definición (Lógica Matemática)

La Lógica Matemática (Lógica Formal, Lógica Simbólica) es la ciencia que estudia la *validez formal* de los **razonamientos**.

- *Razonamiento* (deducción, inferencia, argumentación)

  Obtención de nuevo conocimiento (*conclusión*) a partir de una serie de conocimientos previos (*premisas*).

- *Validez formal*

  Un razonamiento es formalmente válido cuando, suponiendo que todas sus premisas son verdaderas, la conclusión es necesariamente verdadera (es válido en virtud de su forma -su estructura-, independientemente del conocimiento concreto del que se trate). Cuando esto ocurre se dice que la conclusión es una *consecuencia lógica* (*se deduce*) de las premisas.

Para conseguir su cometido la Lógica Matemática dispone de:

Lenguajes Lógicos

Permiten formalizar los razonamientos:

- Su **sintaxis** describe los elementos básicos de los lenguajes y las reglas que permiten construir las expresiones admitidas por ellos, denominadas *términos* (que sirven para representar objetos) y *fórmulas* (que permiten expresar hechos relativos a objetos).
- Su **semántica** permite asignar un significado (valor de verdad, cierto o falso) a las fórmulas, y definir qué significa que un razonamiento sea válido.

Sistemas de demostración

Son sistemas formales que permiten averiguar, mediante la aplicación de una serie de *reglas de inferencia*, si un razonamiento es válido.

Ejemplo (La mortalidad de Sócrates)

Dadas las premisas "*Todas las personas son mortales*" y "*Sócrates es una persona*", ¿se deduce lógicamente "*Hay alguien mortal*"?

1. *Formalización del razonamiento* (en Lógica de Predicados -o Lógica de Primer Orden, LPO-, uno de los lenguajes lógicos más habituales):

Premisa 1:	$\forall X(persona(X) \rightarrow mortal(X))$
Premisa 2:	$persona(sócrates)$
Posible conclusión:	$¿\exists Y\ mortal(Y)?$

2. *Estudio de la validez del razonamiento*: en caso de que las dos premisas sean ciertas, ¿lo es necesariamente también la conclusión? La respuesta es "sí", y se obtiene aplicando a las fórmulas anteriores las reglas de inferencia de cualquiera de los sistemas de demostración existentes para la Lógica de Predicados (sistema de Hilbert, deducción natural, resolución, tableaux semánticos, ...).

Muy breve historia de la Lógica Matemática

- La Lógica fue introducida en el marco de la filosofía por el filósofo griego **Aristóteles** (384-322 A.C).
- El matemático alemán **Leibniz** (siglo XVII) fue el primero en plantear la formalización de la lógica como cálculo matemático, pero esto no se completó hasta mediados del siglo XIX con los trabajos, entre otros, de los matemáticos ingleses **Boole** y **De Morgan**, que aplicaron a la Lógica métodos algebraicos.
- El gran desarrollo de la Lógica Formal se produjo a finales del siglo XIX y primera mitad del XX, con las aportaciones, entre otros, de **Frege, Russell y Whitehead, Hilbert, Herbrand, Church, Turing, Gödel, ...**
- A partir de mediados del siglo XX una parte importante de la investigación en Lógica se centra en el estudio de sus aplicaciones en Informática, en particular como herramienta de programación.

Definición (Demostración Automática)

La Demostración Automática estudia la construcción de sistemas de demostración lógicos que sean adecuados para ser aplicados de forma mecánica y que se puedan *ejecutar de forma eficiente en un ordenador*.

- La Demostración Automática (o Demostración Automática de Teoremas) surge alrededor de 1950, con la aparición de los primeros ordenadores.
- Su objetivo es implementar sistemas de demostración lógicos en los ordenadores, de forma que estos sean capaces de decidir cuándo un razonamiento es válido (convirtiendo así a los ordenadores en máquinas capaces de razonar, y por lo tanto ¡inteligentes!).

- La mayoría de los sistemas de demostración lógicos existentes por aquel entonces resultan poco apropiados para ser implementados en un ordenador y tremendamente ineficientes.

- La investigación en Demostración Automática consigue que en 1960 ya se disponga de sistemas de demostración mucho más eficientes que los anteriores (**Gilmore, Davis-Putnam, ...**).

- El cambio cualitativo se produce en 1965 con la aparición del **Sistema de Resolución con unificación** de **Robinson**: sistema muy eficiente, sencillo y de fácil implementación, adoptado pocos años después por el primer lenguaje de Programación Lógica, el lenguaje PROLOG.

DESCRIPCIÓN Y FUNCIONAMIENTO

Definición (Programación Lógica)

La Programación Lógica es un paradigma de programación que se basa en el uso de la Lógica Matemática y la Demostración Automática como **herramientas de programación**.

- Fue introducida a principios de la década de 1970 por, entre otros, **Greene**, **Kowalski** y **Colmerauer**.
- Se enmarca dentro del campo de la Inteligencia Artificial (cuyo objetivo es la construcción de productos informáticos capaces de reproducir comportamientos "inteligentes").
- Supone un paso más allá respecto a la Demostración Automática: surge de la idea de que la Lógica no sólo se puede usar en un ordenador para demostrar la validez de razonamientos (respuesta "sí o no"), sino también para computar.

- Todo entorno de Programación Lógica consta de los dos siguientes componentes:
  1. Un lenguaje lógico que permite representar el conocimiento.
  2. Un sistema de Demostración Automática para resolver los problemas y computar las soluciones.
- Existen por ende distintos sistemas de Programación Lógica, dependiendo tanto del lenguaje utilizado para representar el conocimiento como del mecanismo de Demostración Automática elegido para computar.
- El primer lenguaje de Programación Lógica, PROLOG, se implementó en la Universidad de Marsella en 1972 por un equipo dirigido por **Colmerauer**.

- En los años 80 aparecen otros lenguajes de Programación Lógica, así como los primeros libros y las primeras implementaciones comerciales de PROLOG.
- Desde entonces el paradigma de la Programación Lógica sigue evolucionando, apareciendo distintas variantes y mezclándose a menudo con otros paradigmas de programación. Surgen así, por ejemplo, la programación lógica concurrente, la programación lógico-funcional, la programación lógica con restricciones, la integración con sistemas orientados a objetos, etc.
- En 1995 se define el estándar ISO-PROLOG, basado en la implementación de la Universidad de Edimburgo (el estándar *de facto* hasta ese momento).

Esquema de funcionamiento

La resolución de un problema mediante Programación Lógica requiere los tres siguientes pasos:

PASO 1: escritura de un programa lógico

El conocimiento relativo al problema que se pretende resolver (el **qué**, no el *cómo*) se representa por medio de una serie de fórmulas lógicas, escritas en el lenguaje lógico asociado con el lenguaje de programación, que constituyen lo que se denomina un programa lógico.

programa lógico = conjunto de fórmulas lógicas

Este paso es **responsabilidad de la programadora o programador**.

PASO 2: escritura de una consulta

El problema a resolver se representa mediante una fórmula lógica, denominada consulta, involucrando a los predicados definidos en el programa lógico.

consulta = fórmula lógica

Este paso es **responsabilidad de la programadora o programador**.

PASO 3: resolución del problema

El problema se resuelve aplicando sobre programa y consulta el sistema de *Demostración Automática* asociado con el lenguaje de programación, con el siguiente doble objetivo:

1. Averiguar si la consulta expresada en el PASO 2 es una consecuencia lógica del conjunto de fórmulas que conforman el programa del PASO 1 (cuyas fórmulas actúan como premisas del razonamiento, mientras que la consulta actúa como posible conclusión).
2. Si efectivamente la consulta resulta ser una consecuencia lógica del programa y es además de tipo *existencial* (*¿Existen objetos X,Y, ... tales que ...?*), computar los valores que hacen que la consulta se deduzca del programa.

Este paso es **responsabilidad del lenguaje de programación**.

Ejemplo

El ejemplo de la página 7, desde el punto de vista de la Programación Lógica y usando la Lógica de Predicados como base:

PASO 1. El programa lógico estaría compuesto por las dos fórmulas

$$\forall X(persona(X) \rightarrow mortal(X))$$
$$persona(sócrates)$$

PASO 2. La consulta sería la fórmula existencial

$$\exists Y\ mortal(Y)$$

PASO 3. Resolución del problema:

Se obtendría una respuesta afirmativa (ya que la consulta es una consecuencia lógica de las dos fórmulas que componen el programa), con el cómputo $Y = sócrates$.

CARACTERÍSTICAS Y APLICACIONES

Una de las características fundamentales de la Programación Lógica, que marca además la diferencia con la Programación Imperativa, es la forma en la que implementa la conocida como **Ecuación de Kowalski**:

Ecuación de Kowalski

$$\boxed{\text{ALGORITMO} = \text{LÓGICA} + \text{CONTROL}}$$

Con la ecuación anterior, Kowalski establece que todo algoritmo consta de dos componentes:

1. Una componente lógica, que describe el conocimiento del que se dispone acerca del problema a resolver (**qué**).
2. Una componente de control, que describe la estrategia que se va a adoptar para solucionar el problema (**cómo**).

Ecuación de Kowalski y paradigmas de programación

- La Programación Imperativa no separa las dos componentes.
- La Programación Lógica las separa claramente:
  - De la componente lógica se responsabiliza el/la programador/a (escribiendo el programa lógico y la consulta).
  - De la componente de control se ocupa el lenguaje de programación (aplicando su sistema de Demostración Automática).

Otras características de la Programación Lógica

- La Programación Lógica prescinde de elementos típicos de otros paradigmas como declaraciones de tipos y variables, vectores, registros, subrutinas, bucles o instrucciones de asignación, y hace un uso extenso de la recursión.
- Los programas lógicos están más próximos a la descripción real del problema que pretenden resolver, por lo que en general son más sencillos, más fáciles de entender, mantener y verificar y más fiables.

Ejemplo (Programa “Abuelas/os”, 1 de 4)

Suponga que se tiene información sobre una serie de pares (progenitor/a, hijo/a) y se necesita un programa que averigüe, a partir de esa información, quiénes son los nietos/as de alguien.

Con Programación Imperativa:

Para escribir un programa imperativo que resuelva el problema anterior, se necesita decidir primero cómo representar la información (vector de registros, vector de vectores, ...) y después qué estrategia usar para computar las/os nietas/os de alguien. Hay varias opciones:

1. Buscar los hijos/as de la persona dada y, a continuación, los hijos/as de estos últimos.
2. Para cada persona, comprobar si alguno de sus progenitores es hija o hijo de la persona dada.
3. ....

Ejemplo (Programa “Abuelas/os”, 2 de 4)

Con Programación Lógica (usando Lógica de Predicados):

PASO 1. Escritura de un programa lógico:

La información se representa mediante *predicados* que establecen *propiedades* (si solo tienen un argumento) o *relaciones* entre sus argumentos (si son varios). En este caso serían necesarios los dos siguientes predicados:

- Predicado *progenitor*(X, Y), cierto si X es progenitor (madre o padre) de Y.
- Predicado *abuelo*(X, Y), cierto si X es abuela o abuelo de Y.

El programa lógico *declara* el conocimiento que se tiene, definiendo los predicados anteriores mediante las *fórmulas lógicas* necesarias.

Ejemplo (Programa “Abuelas/os”, 3 de 4)

- El predicado *progenitor*(X, Y) se define estableciendo las relaciones de progenitura que se conozcan mediante las siguientes fórmulas (en este caso fórmulas atómicas):

$$progenitor(pepa, pepito)$$
$$progenitor(pepito, pepita)$$
$$progenitor(pepito, pepón)$$

- El predicado *abuelo*(X, Y) se define declarando qué significa ser abuela/o mediante la fórmula condicional “para cualesquiera X, Y y Z, si X es progenitor de Y y ese Y a su vez es progenitor de Z, entonces X es abuela/o de Z”:

$$\forall X \forall Y \forall Z \Big((progenitor(X, Y) \wedge progenitor(Y, Z)) \rightarrow abuelo(X, Z) \Big)$$

Ejemplo (Programa “Abuelas/os”, 4 de 4)

PASO 2. Escritura de una consulta:

Para averiguar quiénes son los nietos de, por ejemplo, “pepa”, se establece la siguiente fórmula lógica, correspondiente a la pregunta “¿existe algún N tal que se cumpla la relación $abuelo(pepa, N)$?”:

$$\exists N \; abuelo(pepa, N)$$

PASO 3. Resolución del problema:

Ante el programa y la consulta anterior se obtiene una respuesta afirmativa (ya que la consulta es una consecuencia lógica de las fórmulas que componen el programa), con dos posibles cómputos:

Primera solución: $N = pepita$

Segunda solución: $N = pepón$

Programación Lógica versus Programación Funcional

- La Programación Lógica se basa en el uso de predicados, que establecen *propiedades o relaciones* entre objetos y son por ello más generales que las funciones de la Programación Funcional (toda función puede verse como un caso particular de predicado relacionando los argumentos de entrada de la función con su única salida).
- Los sistemas de Demostración Automática incorporan un mecanismo de búsqueda con retroceso que permite computar todas las posibles soluciones a un problema dado.
- En Programación Lógica los argumentos de los predicados pueden usarse a menudo en modos diferentes, dependiendo de si se facilita un valor concreto -modo entrada- o una variable -modo salida-. Por ello, los programas son más versátiles, puesto que sirven para varios propósitos distintos.

Ejemplo (Capacidad de búsqueda y versatilidad, 1 de 2)

El programa “Abuelas/os” del ejemplo anterior, además de para averiguar quiénes son las/os nietas/os de una persona dada, puede usarse, *sin necesidad de introducir ninguna modificación*, con consultas con otros propósitos distintos, como las siguientes:

1. Para averiguar quiénes son los abuelos de alguien, por ejemplo:

$$\exists A\ abuelo(A, pepón) \quad \text{o bien} \quad \exists A\ abuelo(A, pepito)$$

2. Para encontrar todas las parejas <abuela/o, nieta/o>:

$$\exists A \exists N\ abuelo(A, N)$$

3. Para comprobar si dos personas están relacionadas, por ejemplo:

$$abuelo(pepa, pepito), \quad \text{o bien} \quad progenitor(pepa, pepito)$$

Ejemplo (Capacidad de búsqueda y versatilidad, 2 de 2)

El sistema de Demostración Automática comprueba si las consultas dadas son o no consecuencia lógica del programa y, en caso afirmativo, busca *todas* las posibles soluciones (valores de las variables cuantificadas existencialmente, en caso de que las haya).

Así, las respuestas asociadas a las consultas anteriores, dadas las premisas que conforman el programa, serían:

1. $A = pepa$ (*pepón* solo tiene una abuela, *pepa*) y *false* (*pepito* no tiene abuelas/os).
2. Primera solución: $A = pepa, N = pepita$
   Segunda solución: $A = pepa, N = pepón$
3. *false* (pepa no es abuela de pepito) y *true* (pepa sí es progenitora de pepito)

Aplicaciones más comunes de la Programación Lógica

Aunque los lenguajes de Programación Lógica son lenguajes de programación de *propósito general*, resultan más adecuados para la resolución de problemas simbólicos que para la resolución de problemas numéricos. Algunos de los campos más destacados en los que se usa este paradigma son:

- Inteligencia Artificial, en particular problemas de planificación y búsqueda.
- Sistemas expertos, sistemas basados en el conocimiento.
- Reconocimiento y procesamiento de lenguaje natural.
- Sistemas de recomendación y asesoramiento.
- Manejo y mantenimiento de bases de datos.
- Análisis de redes sociales, Web Semántica.
- ...

BIBLIOGRAFÍA

- L. Sterling and E. Shapiro. *The Art of Prolog*. The MIT Press, Cambridge, Mass., second edition, 1994.
- W.F. Clocksin and C.S. Mellish. *Programming in Prolog*. Springer-Verlag, Berlin, fifth edition, 2003.
- I. Bratko.*Prolog Programming for Artificial Intelligence*. Addison-Wesley, Reading, Massachusetts, third edition, 2001.
- J. Lloyd. *Foundations of Logic Programming*, (Second Edition). Springer-Verlag, 1987.
- R. O'Keefe. *The Craft of Prolog*. The MIT Press, Cambridge, MA, 1990.
- U. Nilsson and J. Maluszynski. Logic, Programming and Prolog. John Wiley & Sons Ltd, 1996.
- SWI-Prolog, entorno de programación en Prolog de dominio público.
- comp.lang.prolog. Faq