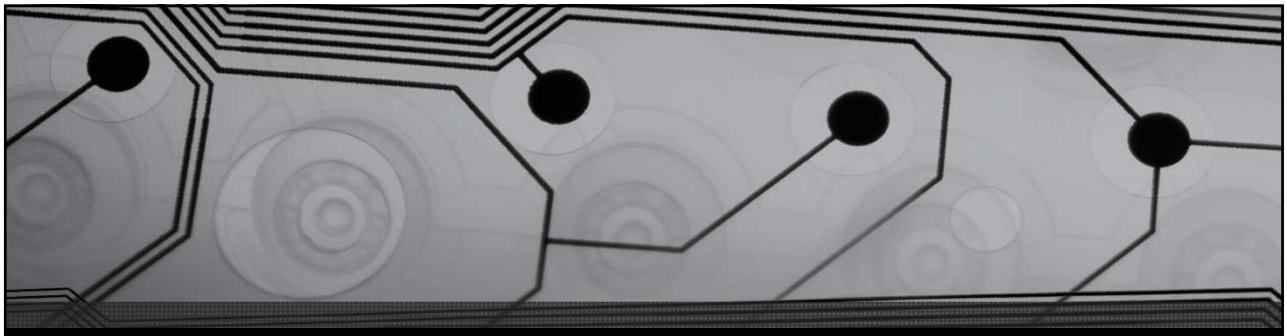
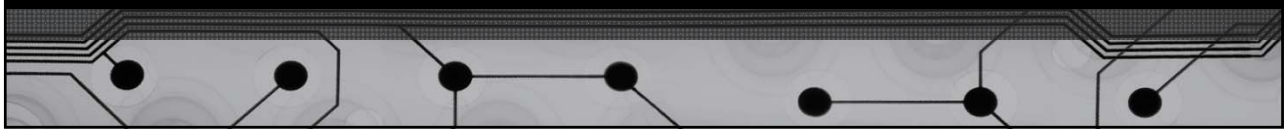


Programación Lógica

Dr. Víctor de la Cueva
vcueva@itesm.mx



Introducción

- El concepto de programación lógica está ligado históricamente a un lenguaje de programación llamado Prolog.
- Prolog fue desarrollado en 1972 y hasta ahora es el único lenguaje de su clase cuyo uso se ha extendido.
- Prolog se aplicó en principio al procesamiento del lenguaje natural, desde entonces se ha utilizado para: especificar algoritmos, búsquedas en bases de datos, escritura de compiladores, construcción de sistemas expertos, etc.
- Prolog es muy adecuado para aplicaciones que implican búsqueda de patrones, búsqueda con backtracking o información incompleta.
- Prolog (la herramienta o el lenguaje) lleva a la práctica la programación lógica (el concepto).

Cálculos con relaciones

- La programación lógica trabaja con relaciones más que con funciones.
- Se basa en la premisa de que programar con relaciones es más flexible que programar con funciones, debido a que las relaciones tratan de modo uniforme a los argumentos y a los resultados.
- De manera informal, las relaciones no tienen sentido de dirección ni prejuicio alguno acerca de qué se calcula a partir de qué.

Listas en Prolog

- Vamos a presentar Prolog más adelante pero podemos anticipar su notación de listas.
- Las listas se escriben entre corchetes [y].
- [] es la lista vacía.
- [a,b] es una lista de dos símbolos, a y b.
- Si H es un símbolo y T una lista, entonces [H | T] es una lista con *head* H y *tail* T, por lo que:

[a, b, c] = [a | [b,c]]

Relaciones

- Una visión concreta de cualquier relación es una tabla con n≥0 columnas y un conjunto posiblemente infinito de renglones.
- La tupla $(a_1, a_2, \ldots, a_n)$ está en una relación si a_i aparece en la columna i, $1 \leq i \leq n$, de algún renglón de la tabla de relación.
- El ejemplo ejecutable de esta sección es la relación `agrega` (se define en el siguiente slide) sobre listas.
- Las relaciones se conocen también como predicados.
- Un nombre de relación `rel` puede verse como una prueba de la forma: ¿se encuentra una tupla dada en la relación `rel`?

Relación agrega

- La relación `agrega` es un conjunto de tuplas de la forma (X, Y, Z), donde Z consiste en los elementos de X seguidos de los de Y.
- Algunas de las tuplas de `agrega` son:

agrega		
X	Y	Z
[]	[]	[]
[a]	[]	[a]
[a,b]	[c,d]	[a,b,c,d]

Reglas y hechos

- Las relaciones se especifican por medio de reglas que se escriben en pseudocódigo, como:

$$P \textbf{ if } Q_1 \textbf{ and } Q_2 \textbf{ and } \ldots \textbf{ and } Q_k \qquad \text{para } k \geq 0$$

  - Tales reglas se llaman Cláusulas de Horn [1951], en honor de Alfred Horn (1918-2001), quien las propuso y las estudió.
  - Los lenguajes han optado por trabajar con cláusulas de Horn debida que llevan a implementaciones eficientes.
- Observe que una cláusula de Horn tiene la forma:

$$\textbf{if } Q_1 \textbf{ and } Q_2 \textbf{ and } \ldots \textbf{ and } Q_k \textbf{ then } P$$

- Un hecho es un caso especial de una regla en el cual k=0 y P no almacena ninguna condición. Se escribe simplemente como: P.

Fuente de foto: http://e-ducation.datapeak.net/mathematicians.htm

Ejemplo con agrega

- La relación `agrega` se especifica mediante dos reglas.
- La primera es un hecho que establece que las ternas de la forma ([], Y, Y) se encuentran en la relación `agrega`. En pseudocódigo:

agrega [] y Y para obtener Y

- La segunda, usa la notación [H | T]:

agrega [H | X_1] y Y para obtener [H | Z_1] **if**

agrega X_1 y Y para obtener Z_1

- De la regla se tiene que:

agrega [a,b] y [c,d] para ,obtener [a,b,c,d] **if**

agrega [b] y [c,d] para obtener [b,c,d]

Ejemplo con agrega (cont.)

- Aquí H = a, X_1 = [b], Y = [c,d] y Z_1 = [b,c,d].
- Observe que [a | [b]] es la misma lista [a,b] y que [a | [b, c, d]] es la misma lista que [a,b,c,d].

NOTAS

- P, Q_1, Q_2, …,Q_k son términos.
- Un término es una constante o una variable o tiene la forma rel(T_1,T_2,…,T_n) para n≥0, donde rel es el nombre de la relación y T_1,T_2,…,T_n son términos.
- Por convención, los nombres de las variables comienzan con MAYÚSCULA; los nombres de las constantes y las relaciones comienzan con minúscula.

Consultas

- La PL está dirigida por consultas acerca de las relaciones.
- La consulta más simple pregunta si determinada tupla pertenece a una relación. Ej

agrega [a,b] y [c,d] para obtener [a,b,c,d]?

Respuesta: Sí

Pregunta en PL es si la terna ([a,b],[c,d],[a,b,c,d]) pertenece a la relación agrega.

Consultas negativas

- Las cláusulas de Horn no pueden representar información negativa, es decir, no podemos preguntar directamente si una tupla no se encuentra en una relación.
- Las consultas de este tipo tendrán respuestas sí/fracaso en lugar de sí/no.
- Por "fracaso" se trata de decir que se fracasó en la deducción de una respuesta "sí".

Consultas con variables

- Las consultas que contienen variable son más interesantes.

  Hay una Z tal que

  agrega [a,b] y [c,d] para obtener Z?

  Respuesta: sí, cuando Z = [a,b,c,d]

- Lo que parece ser una consulta sí/fracaso es en realidad una petición de valores adecuados para las variables que se encuentran en la consulta.
- La consulta anterior es la petición de una Z tal que la terna ([a,b],[c,d],Z) esté en la relación agrega.

Flexibilidad de las relaciones

- Un beneficio de trabajar con relaciones es que si agregamos X a Y para obtener Z, entonces X,Y o Z pueden calcularse a partir de las otras dos.
- Esta propiedad se refiere al comentario anteriormente hecho de que las relaciones son flexibles debido a que no tienen prejuicios acerca de qué se calcula a partir de qué.
- X puede calcularse a partir de Y y de Z:

  Existe una X tal que

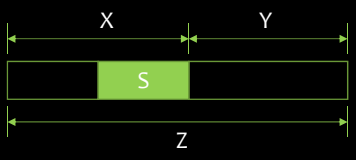
  agrega X y [c,d] para obtener [a,b,c,d]? Respuesta: sí, cuando X = [a,b]
- Y puede calcularse a partir de X y de Z:

  Existe una Y tal que

  agrega [a,b] y Y para obtener [a,b,c,d] Respuesta: sí, cuando Y = [c,d]

Relaciones nuevas a partir de anteriores

- Las relaciones nuevas pueden definirse a partir de las anteriores.
- En las siguientes tres reglas para `prefijo`, `sufijo` y `sublista`, las variables X, Y y Z hacen referencia a las partes de una lista:



prefijo X de Z **if**

para algunos Y, agrega X y Y para obtener Z.

subfijo Y de Z **if**

para algunos X, agrega X y Y para obtener Z.

sublista S de Z **if**

para algunos X, prefijo X y Z **and** subfijo S de X.

¿Qué es la Programación Lógica?

- El término Programación Lógica nos remite de manera vaga a:
  - El uso de hechos y reglas para representar información, y
  - El uso de deducciones para responder consultas.
- Nosotros como programadores proporcionamos los hechos y las reglas (lógica) mientras que el lenguaje usa la deducción (control) para calcular respuestas a consultas.
- El control se relaciona con la forma en la que pueden aplicarse las reglas en un orden particular.
- Normalmente, el control de Prolog procede de izquierda a derecha (cada dialecto tiene sus propias nociones de control).

El control

La regla

$$P \textbf{ if } Q_1 \textbf{ and } Q_2 \textbf{ and } \ldots \textbf{ and } Q_k \quad k \geq 0$$

puede leerse como:

para deducir P,

deduce Q_1;

deduce Q_2;

...

deduce Q_k;

- Esta sencilla estrategia es sorprendentemente versátil y flexible. Por desgracia, algunas veces se atora en ciclos infinitos y puede producir anomalías que implican negaciones.

Referencias

- R. Sethi. Programming Languages: concepts and constructs. Addison-Wesley, 2nd edition (1996).