PROGRAMACIÓN LÓGICA

I. Sistemas para la programación lógica.
II. Programación elemental con bases de datos y programación recursiva.
III. Programación con listas y árboles.
IV. Control explícito en Prolog.
V. Metaprogramación.

Entorno y herraminetas

SWI-Prolog

entorno de programación Prolog

http://www.swi.psy.uva.nl/projects/SWI-Prolog/

SLD-Draw

representación de árboles SLD

http://www.lcc.uma.es/~pacog/sldDraw

Bibliografía

"The Art of Prolog" (2ed)
Leon Sterling y Ehud Shapiro
MIT Press, 1994

"Prolog Programming for Artificial Intelligence" (3ed)
Ivan Bratko
Addison-Wesley, 2000

"Clause and Effect"
William F. Clocksin
Springer-Verlag, 1997

"Programación en Prolog" (2ed)
William F. Clocksin y Chris S. Mellish
Gustavo Gili, 1993

Bibliografía en internet

Libros gratuitos disponibles en internet:

"Prolog Programming: A First Course"
Paul Brna
http://cbl.leeds.ac.uk/~paul/prologbook/

"Logic, Programming and Prolog" (2ed)
Ulf Nilsson y Jan Maluszynski
http://www.ida.liu.se/~ulfni/lpp/

Sistema para la Programación Lógica.

- I.1 Términos, relaciones y programas.
- I.2 Interpretación declarativa de un programa.
- I.3 Interpretación operativa de un programa.
- I.4 Resolución SLD.
- I.5 Unificación de términos e instanciación de variables.

Términos, relaciones y programas.

Términos

– En los programas lógicos se opera con **objetos** o **términos funcionales**:

constantes -denotaciones de objetos simples;
– (números o funtores sin parámetros)

estructuras -denotaciones de objetos compuestos;
– (funtores con parámetros)

Estructuras simples(registros).

Estructuras recursivas.

variables - denotaciones genéricas de objetos
» (simples y compuestos)

Ejemplo de términos

Constantes

cero, 1.3, antonio, casado, soltero, nil, arbol_vacio

(los nombres de funtores comienzan con minúscula)

Estructuras simples(registros):

ficha(antonio, casado) ,
substancia(azufre,16,32.1).
arco(a,b)

Estructuras recursivas:

suc(cero), suc(suc(cero)), ...
arbol(antonio,arbol_vacio,arbol_vacio),
arbol(antonio,arbol(juan,arbol_vacio,arbol_vacio),arbol_vacio)

Variables

X, Nombre, Ficha, Estado_civil

(los nombres de variables comienzan con mayúscula)

Relaciones/Predicados

- Representan correspondencias entre términos funcionales
  - Se utilizan para construir términos predicativos o fórmulas atómicas susceptibles de satisfacción o valoración lógica.

```
padre(antonio, jose)
elemento(3,[1,5,7])
```

Son los sujetos de los cálculos en la programación lógica.

No son objetos de cálculo.

Definición de relaciones

- Mediante cláusulas de Horn
  - *estableciendo hechos*
    fórmula-atómica "."
  - *estableciendo reglas*
    fórmula-atómica ":-" fórmula-atómica {","fórmula-atómica}"."
- *La definición de una relación puede requerir de varias cláusulas*
- *Cada relación se caracteriza por su nombre y su número de argumentos o aridad.*

Ejemplo

```
padre(antonio, maria).
padre(antonio, luis).                % padre/2

madre(maria, jose).                  % madre/2

persona(X).                          % persona/1

progenitor(A,D) :- padre(A,D).
progenitor(A,D) :- madre(A,D).       % progenitor/2

antepasado(A,D) :- progenitor(A,D).
antepasado(A,D) :- progenitor(A,H),
                   antepasado(H,D).  % antepasado/2
```

Sintaxis de Prolog

```
cláusula::= fórmula-atómica "."
         | fórmula-atómica ":-" fórmula-atómica {","fórmula-atómica}"."
fórmula-atómica ::= predicado "(" término {"," término} ")"
predicado ::=     letra-minúscula {letra | dígito}
término ::=       constante | estructura | variable
constante ::=     letra-minúscula {letra | dígito}  |   número
estructura ::=    funtor "(" término {"," término}  ")"  | lista
funtor ::=        letra-minúscula {letra | dígito}
lista ::=         "[]"  |   "["cabecera "|" cola "]"
cabecera ::=       término
cola ::=           lista
variable ::=       letra-mayúscula {letra | dígito}
                     | "_"{letra | dígito}
```

Programa

Definiciones de relaciones
(Base de conocimiento)
+
Cuestión

Cuestión/Objetivo

- Conjunción de fórmulas atómicas establecidas entre objetos que pueden ser fijos, variables o estar determinados parcialmente.

```
"?-"fórmula-atómica {"," fórmula-atómica}"."
```

Cuestiones

- Cuestiones de mera comprobación

  ?- padre(antonio, maria).
  ?- padre(antonio, jose).

  (se aplica la *identidad* como regla de deducción)

  ?- persona(antonio).

  (se aplica la *instanciación* como regla de deducción:
  de una proposición cuantificada universalmente
  se deduce cualquier instancia)

Cuestiones

Cuestiones para el cálculo de valores de objetos

?- padre(antonio, X).
?- padre(X, maria).
?- progenitor(antonio, X), madre(X, jose).

(se aplica la *generalización* como regla de deducción:
una cuestión existencial es siempre una consecuencia lógica de la existencia de una instancia de dicha cuestión)

Interpretación declarativa

Cláusulas de Horn

Las cláusulas de Horn son cláusulas con, a lo más, un literal positivo

$$\forall X_1,...,X_n \bullet (A \vee \neg B_1 \vee ... \vee \neg B_k)$$

- Una cláusula de la forma

  A .

  sólo tiene el literal positivo y se interpreta simplemente como la afirmación de que

  la fórmula A es cierta, para todos los valores posibles de las variables que aparecen (si aparece alguna).

Interpretación declarativa

- Una cláusula completa

  $A :- B_1, B_2, ..., B_k.$

  se interpreta como una implicación de la forma siguiente:

  $$A \Leftarrow B_1 \wedge B_2 \wedge ... \wedge B_k$$

  con todas las variables que aparezcan cuantificadas universalmente:

la fórmula A es cierta, para todos los valores posibles de las variables que aparezcan, para los que son ciertas todas las fórmulas B_i; lo que puede requerir la existencia de alguna combinación de valores para las variables que no aparecen en A .

Interpretación declarativa

```
antepasado(A,D) :-
    progenitor(A,H),antepasado(H,D).
```

se interpreta como

$$\forall A, D, H \bullet \left(progenitor(A, H) \wedge antepasado(H, D) \Rightarrow antepasado(A, D)\right)$$

```
arco(a,b).
arco(a,d).
arco(b,c).
camino(X,Y) :- arco(X,Y).
camino(X,Y) :- arco(X,Z), camino(Z,Y).
```

Interpretación declarativa

Una cuestión

?- B_1, B_2, ... , B_k .

sólo tiene literales negativos y se interpreta como

para cada combinación de valores de las variables que aparecen, alguna fórmula B_i no es cierta

?- progenitor(antonio, X), madre(X, jose).

significa $\forall X \bullet (\neg progenitor(antonio, X) \vee \neg madre(X, jose))$

si conseguimos una refutación, habremos probado lo contrario

$$\neg\forall X \bullet (\neg progenitor(antonio, X) \vee \neg madre(X, jose))$$

o lo que es lo mismo

$$\exists X \bullet (progenitor(antonio, X) \wedge madre(X, jose))$$

Interpretación operativa de un programa

- **Cláusulas de Horn** (interpretación procedimental)

  Una cláusula de Horn $\boldsymbol{A :- B_1, B_2, ..., B_k}$. se puede interpretar también en la forma

  para resolver A se debe resolver B_1, B_2 ,...y B_k ;

  con lo que será comparable a la definición de un procedimiento, donde

  A hace el papel de la **cabecera** del procedimiento,

  B_1, B_2,...y B_k serán el **cuerpo**, interpretado como una serie de llamadas a procedimientos auxiliares B_i ,

Resolución SLD

Selective Lineal for Definite clauses

Procedimiento de refutación

?- $B_1, B_2, ..., B_k$.

Cada **paso de resolución**, aplicado a una cuestión o *resolvente* ?- $B_1, B_2, ..., B_k$. consiste en la siguiente secuencia de operaciones:

r1) **seleccionar una fórmula** B_i del resolvente,

r2) **seleccionar una cláusula** *C* del programa, dentro del procedimiento correspondiente al predicado que aparece en B_i, cuya cabecera coincida con B_i (en realidad serán literales opuestos),

r3) si se encuentra tal cláusula, sustituir, en el resolvente, B_i por el cuerpo de *C*; si no la resolución termina sin éxito.

ejc00.pl

Situaciones de indeterminación

- Las reglas para la selección de fórmulas dentro de una cuestión se denominan ***reglas de cálculo*** o ***de selección*** y se demuestra que no afectan al resultado de los cálculos (únicamente afectan al número de pasos).
- Las reglas para la selección de cláusulas de la base de conocimiento se denominan ***reglas de búsqueda*** y sí tienen consecuencias sobre el resultado de los cálculos.

Espacio de búsqueda asociado a un Programa

- Fijada una *regla de cálculo*, los distintos cálculos posibles para un programa constituido por una cuestión *Q* y una base de conocimiento *P* se pueden representar gráficamente mediante un *árbol de búsqueda* caracterizado por:
  - su raíz, etiquetada con la cuestión *Q* o primer resolvente;
  - y los demás nodos etiquetados con los resolventes producidos por pasos de resolución; además
  - para cada nodo etiquetado con un resolvente no vacío, si *A* es la fórmula que selecciona la regla de cálculo, existirán tantos descendientes como cláusulas con cabecera coincidente con *A* existan en el procedimiento de definición del correspondiente predicado,
  - los nodos etiquetados con resolventes vacíos no tienen descendientes.
- Las distintas *reglas de búsqueda* representan distintas formas de recorrido de los árboles de búsqueda.

ejc01.pl

Unificación de términos

```
...
progenitor(A,D) :- padre(A,D).
...
?- progenitor(antonio,X).
```

Sustituimos `antonio` por `A`, y `X` por `D` y tomamos la regla

```
progenitor(antonio,X) :- padre(antonio,X).
```

La nueva resolvente es

```
padre(antonio,X).
```

Sustituciones e Instancias.

Una *sustitución* de variables es un conjunto finito

$\theta = \{V_1/t_1, ...,V_n/t_n\}$

de *ligaduras* V_i/t_i donde cada V_i es una variable distinta y cada t_i un término distinto de V_i.

Una *instancia* de una fórmula *E*, por aplicación de una sustitución θ, es la expresión $E\theta$ obtenida tras la sustitución simultánea de todas las variables que aparecen en θ por los términos de sus correspondientes ligaduras;

p.e.: si `E = p(X, f(Y),a)` y `θ = {X/b, Y/X}`

`Eθ = p(b, f(X), a).`

Sustituciones

- Una *redenominación* o cambio de nombre de las variables de una fórmula *E* es una sustitución de variables que aparecen en la expresión por otras variables que no aparecen en la expresión o por algunas de las que se van a cambiar;

  p.e.: para $E = p(X, f(Y), a)$,

  $$\theta = \{X/Z, Y/X\}$$

  es una redenominación de las variables de E.

Composición de sustituciones

Sean dos sustituciones

$$\theta = \{V_1/t_1, ..., V_n/t_n\} \text{ y } \tau = \{U_1/s_1, ..., U_m/s_m\}$$

Se define como la sustitución resultante de aplicar primero θ y después τ. Esta sustitución es equivalente a la sustitución obtenida a partir de

$$\{V_1/t_1\tau, ..., V_n/t_n\tau, U_1/s_1, ..., U_m/s_m\}$$

eliminando

1) las ligaduras U_i/s_i tales que la variable U_i coincida con alguna variable V_j; y

2) las ligaduras $V_i/t_i\tau$ tales que $V_i = t_i\tau$.

p.e.: la composición de $\theta = \{X/f(Y), Y/Z\}$ y $\tau = \{X/a, Y/b, Z/Y\}$ será

$$\theta\tau = \{X/f(b), Z/Y\}$$

Ejercicios

E=p(X, g(Y), Z)

σ= {X/f(Y)} θ= {Y/a, Z/b, X/c}

Calcular

(E σ) θ

E (σ θ)

Unificadores

Un *unificador* de dos fórmulas atómicas *A* y *B* es una sustitución θ tal que

$$A\theta = B\theta;$$

p.e.: θ ={Y/f(a), X/a, Z/a} es un unificador para las fórmulas

A =p(f(X),Z) y B = p(Y,a),

pues

Aθ = p(f(a), a) = Bθ.

Otro es

θ ={Y/f(g(b)), X/g(b), Z/a}

Unificador de máxima generalidad

Un unificador de dos fórmulas atómicas *A* y *B* se dice que es un *unificador de máxima generalidad* (o un *umg*) cuando cualquier otro siempre se puede expresar como composición de éste con alguna sustitución (viene a ser el que establece el menor número de ligaduras)

p.e.: para A = p(f(X),Z) y B = p(Y,a),
θ = {Y/f(X), Z/a}
es un unificador de máxima generalidad.

Un *umg* de dos términos no siempre es único, pueden existir varios; pero entonces se diferenciarán únicamente en una redenominación de las variables;

p.e.: para A = p(f(X), Z) y B = p(f(Y),V),
θ = {X/Y, Z/V} y τ = {Y/X, Z/V}
son dos *umg*s tales que
θ= τ{X/Y} y τ =θ{Y/X}

Algoritmo de unificación

0) **SI** ambas fórmulas comienzan por predicados distintos,
ENTONCES (no son unificables) *PARAR*
SI NO *CONTINUAR*;
1) k := 0; $\theta_0 = \varepsilon$ (sustitución vacía);
2) **SI** $E\theta_k = F\theta_k$
ENTONCES (θ_k será un umg) *PARAR*
SI NO determinar aquéllos dos términos que comienzan en el primer carácter de $E\theta_k$ y $F\theta_k$, respectivamente en el que difieren ambas expresiones (primer par de discordancia);
3) **SI** uno de los términos es una variable X y el otro es un término t donde no aparece X
ENTONCES construir θ_{k+1} componiendo θ_k con {X/t};
k:=k+1;
IR A 2)
SI NO (no son unificables)
PARAR

Ejercicios

Aplicar el algoritmo de unificación a los pares siguientes

a) p(f(a), g(X)), p(Y,Y)

b) p(a, X, h(g(Z))), p(Z, h(Y),h(Y))

c) p(X,X), p(Y, f(Y))

Resolución con variables

Si E y F son fórmulas atómicas unificables, con umg θ, entonces

$\{(E \vee C)\theta, (\neg F \vee C')\theta\} \models (C \vee C')\theta$

Todas las instancias de $(C \vee C')\theta$ son consecuencia lógica de las instancias

$(E \vee C)\theta$ y $(\neg F \vee C')\theta$

Extensión del algoritmo de resolución

?- $B_1, B_2, ..., B_k$.

r1) seleccionar una fórmula B_i del resolvente;
r2) seleccionar una cláusula C del programa, dentro del procedimiento correspondiente al predicado que aparece en B_i ;
r3) redenominar las variables de la cláusula C, si es necesario, para que no coincidan con las variables que aparecen en el resolvente;
r4) unificar B_i con la cabecera de la cláusula (redenominada);
r5) si la unificación tiene éxito y produce un *umg* θ
- sustituir, en el resolvente, B_i por el cuerpo de C (redenominada),
- aplicar las sustituciones de θ al resolvente resultante para obtener un nuevo resolvente.

ejc02.pl

Resolución SLD aplicada a cláusulas de Horn

- *consistente* -si para un objetivo *Q* existe un cálculo que tiene éxito, dicho objetivo es consecuencia lógica de la base de conocimiento, desde el punto de vista de la semántica declarativa, para los valores de las variables que aparecen en la sustitución calculada; y
- *completa* – si un objetivo *Q* es consecuencia lógica, desde el punto de vista de la semántica declarativa, para ciertos valores de las variables, entonces existe un cálculo de resolución que termina en éxito y produce los mismos valores para las variables

Prolog

- Regla de selección:
  - De izquierda a derecha
- Regla de búsqueda:
  - De arriba abajo
- ¿Cómo construye el árbol SLD?
  - En profundidad de izquierda a derecha
    - Backtraking o reevaluación
    - Esto hace que no sea completo

ejc01.pl

Ejemplos

```
p(X,Y):-q(X,Y).
p(X,Y):-q(a,X).
q(a,a).
q(X,a):- r(Y),s(X,Y).
q(X,Y):-r(X),p(X,Y).
s(b,b).
s(b,X):-r(X).
r(b).
r(a).
```

```
?-p(b,Z).
```

ejc09.pl

Ejemplos

```
p(X,v,v).
p(X,f(X,Y),Z):-p(X,Y,Z).
p(X,f(Y,Z),f(X,T)):-p(X,Z,T).

                ?-p(X,f(a,f(b,f(a,v))),L).
```

ejc10.pl