Programación Lógica

Miguel Álvarez
Juan Saab
Óscar Suárez
Fredy Virguez

Mayo 27 de 2015
Lenguajes de Programación.
Universidad Nacional de Colombia.

Agenda

- Filosofía.
- Conceptos claves.
- Ventajas y desventajas.
- Lenguajes de Programación.
- Ejemplos en distintos lenguajes.
- Aplicaciones de este paradigma.
- Infografía.

Paradigma Lógico.

La filosofía del paradigma lógico se basa en la aplicación de las reglas de la lógica para inferir conclusiones a partir de datos.

Programación Lógica.

- ★ Es el punto donde confluyen la lógica y la informática.
- ★ Es aquella en la que se usa el **razonamiento lógico**, automatizado computacionalmente; para modelar el conocimiento sobre un problema.

Programación Lógica.

★ Según Robert Kwalski la programación logica es:
algoritmo= logica + control.

Lógica(Programador):hechos y reglas que representan el conocimiento.
Control(Intérprete):Deducción lógica para dar respuestas.

Programación Lógica

- ★ Un programa en un lenguaje lógico es una descripción específica de un problema.
- ★ Un problema es descrito vía axiomas y reglas de deducción.
- ★ Ejecutar un programa lógico consiste en determinar si una conclusión es verdadera o falsa en base a las sentencias lógicas especificadas.

Programación Lógica: Principales Rasgos.

- **Programa:**Reglas Lógicas.
- **Operacional**:calusulas de *Horn*.
- **Declarativa:**
  - lógica de primer orden.
  - uso múltiple de un procedimiento.
  - variables lógicas
  - datos parcialmente especificados.
  - no evaluación perezosa.
  - no estructuras infinitas.

Conceptos Clave.

- **Lógica de primer orden.**
- **Términos**(Constantes y Variables).
- **Predicados**
- **Cuantificadores.**
- **Fórmulas.**
- **Cláusulas de Horn.**

Conceptos Clave.

La lógica de primer orden:

Usa a las frases declarativas simples como elementos base para modelar problemas;cuenta con un conjunto de reglas de inferencia para evaluar la veracidad o derivar nuevo conocimiento.

Conceptos Clave.

“los ingenieros son listos, Juan es ingeniero por lo tanto Juan es listo”

$$(\forall x(ingeniero(x) \rightarrow listo(x)) \wedge ingeniero(juan)) \rightarrow listo(juan).$$

Conceptos Claves

Terminos:

Constantes: Representan exactamente un elemento del dominio particular.

- Todo nombre debe referir a un objeto.
- Ningún nombre puede referir a más de un objeto.
- Un objeto puede tener más de un nombre.

Variables: son simbolos que pueden tomar los valores constantes.

Conceptos clave.

Predicados: equivalen a frases u oraciones del lenguaje hablado**,** son utilizados para denotar alguna propiedad de objetos o alguna relación entre objetos.

Juan es padre de Ana.

sujeto: *Juan.*

Predicado: *es padre de Ana.*

es_padre(juan,ana)

Conceptos clave.

Predicados: se representan letras mayúscula del alfabeto latino, con parámetros, preferentemente representados por letras minúsculas del mismo alfabeto a partir de x, entre paréntesis separados por comas.

ejemplo: ***P(x)***

P(x) *es la formalización de: “x es guitarrista”*

donde ***x*** es una variable

Conceptos Claves.

Aridad: es un número que indica cuántas variables individuales necesita el símbolo de predicado para formar una oración.

Oración	Predicado	Aridad
Ana regaló	regalo(Ana)	1
Ana regaló flores	regalo(Ana, flores)	2
Ana regaló flores a Martina	regalo(Ana, flores, MArtina)	3

Conceptos Claves.

Un **predicado** se clasifica según su **ARIDAD (n)** de la siguiente manera:

1. si **n**=0 los predicados son enunciados.
2. si **n**= 1 se denominan propiedades.
3. si **n**= 2 se denominan relaciones.
4. a partir de **n** =3 no existen nombres específicos.

Conceptos Claves.

ejemplos:

- unarios:P(x):
  - “x es una persona”.
  - “ x es de color rojo”.
- binarios:Q(x,y):
  - “El cuadrado de x es y”.
  - “x come y”.
- ternarios:S(x,y,z):
  - “La suma de x y de y es z”.
  - “x trae a y con z”.

Conceptos Claves.

Los **cuantificadores** son los dos operadores usados para indicar cantidad en una proposición.

Símbolo	Nombre	Significado
∀	cuantificador universal	todos los/as… cada...
∃	cuantificador existencial	hay un… existe un...

Conceptos Claves.

Ejemplos con cuantificadores:

Si ***P(x)*** quiere decir “x es un estudiante”, entonces:

- ∃ ***x P(x)*** significa ‘hay estudiantes’, ‘existen estudiantes’, ‘algunos son estudiantes’, ‘alguno es un estudiante’, etc.
- ∀ ***x P(x)*** significa ‘todos son estudiantes’, ‘todo el mundo es estudiante’, etc

Conceptos Clave.

Fórmulas:

El lenguaje de la lógica de predicados se denomina lenguaje de fórmulas.

Este lenguaje utiliza como alfabeto:

Alfabeto{ $\wedge$, $\vee$, $\rightarrow$,$\neg$,$\forall$,$\exists$,*P,Q,R,...,a,b,c,...,x,y,z,(,)*}

Conceptos Claves.

Cláusulas de Horn: llamadas así en honor al lógico Alfred Horn. Estas reglas están compuestas por dos partes:

el consecuente y el antecedente.

El consecuente, es lo que se quiere probar,
El antecedente es la condición que determinará en qué casos el consecuente es verdadero o falso.

Conceptos Claves.

Una **cláusula de Horn,** es un tipo particular de lógica de predicados.

H ← P1,P2,...,Pn.

H es verdadera si y sólo si P1,P2,...,Pn son verdaderas.

Ejemplo: En una ciudad C, está nevando si hay precipitación en la ciudad y la temperatura es muy baja

Nevando(C) ← Precipitación(C), Helando(C)

Ventajas

- Simplicidad
- Cercanía a las especificaciones del problema realizada con lenguajes formales
- Sencillez, potencia y elegancia
- Metodología rigurosa de especificación

Ventajas

- Sencillez en la implementación de estructuras complejas
- Programas más fáciles de depurar y mantener que los lenguajes

Desventajas

- Poco eficientes
- Poco utilizado en aplicaciones reales
- La resolución automática no siempre es eficiente, por lo que eventualmente se podría dar una respuesta incorrecta a una consulta

Desventajas

- Ciertos problemas están ligados a la representación del conocimiento (Prolog)
- Si el programa no contiene suficiente información para contestar una consulta responde no, aunque debería responder que no es posible inferir un resultado (Prolog)

Desventajas

- Los motores de inferencia poseen algunos límites
- Se quedan cortos en portabilidad, riqueza de librerías, interfaces con otros lenguajes y herramientas de depuración

Lenguajes de Programación

- ALF (Another Logical Framework)
  - Programación lógica - funcional (evaluación de expresiones y funciones matemáticas).
  - Cláusulas de Horn.
- Gôdel programming language
  - Polimorfismo.
  - Meta-programación.

Lenguajes de Programación

- Mercury Programming Language
  - Lenguaje de alto nivel
  - Derivado de Prolog
  - Declarativo (conclusiones lógicas)
  - Parámetros: estado anterior y demás.
  - Salida: nuevo estado.

Derivados de Prolog

- Ace, PALS
  - Soporta arquitectura multiprocesador
- Actor Prolog
- CLP(FD)
- CSP (Constraint Satisfaction Problem):
  - Generación de variables.
  - Definición de restricciones.
  - Labeling.

Derivados de Prolog

- Lambda Prolog
  - Tipos polimórficos, módulos y tipos de datos abstractos.
- Logtalk
  - Extensión de Prolog orientada a objetos.

Ejemplos

Mercury

```
:- module hello_world.
:- interface.
:- import_module io.
:- pred main(io__state, io__state).
:- mode main(di, uo) is det.
:- implementation.
main -->
io__write_string("Hello, World!\n").
```

Mercury

```
:- module fib.
:- interface.
:- import_module io.
:- pred main(io::di, io::uo) is
det.
:- implementation.
:- import_module int.
:- pred fib(int::in, int::out) is
det.
```

```
fib(N, X) :-( if N =< 2
then X = 1
else fib(N-1,A),
   fib(N-2,B),
   X=A+B).
   main(!IO) :-fib(17, X),
   io.write_string("fib(17,",
!IO),
   io.write_int(X, !IO),
   io.write_string(")\n", !
IO).
```

Gôdel

```
MODULE      GCD.
IMPORT      Integers.
PREDICATE   Gcd : Integer * Integer * Integer.
Gcd(i,j,d) <- CommonDivisor(i,j,d) &~ SOME [e]
(CommonDivisor(i,j,e) & e > d).
 PREDICATE   CommonDivisor : Integer * Integer * Integer.
CommonDivisor(i,j,d) <- IF (i = 0 \/ j = 0) THEN  d =
Max(Abs(i),Abs(j))                                ELSE      1
=< d =< Min(Abs(i),Abs(j)) &
                          i Mod d = 0 & j Mod d = 0.
```

Actor Prolog

```
project: ((‘Hello’))
class ‘Hello’ specializing ‘Console’
[
goal:-
   writeln(“Hello World!”)
]
```

CLP (FD) *Librería de Prolog

```
:-use_module(library(clpfd)).

n_factorial(0,1).
n_factorial(N,F):-
  N #> 0,
  N1 #= N-1,
  F #= N*F1,
  n_factorial(N1,F1).
```

CLP (FD) *Librería de Prolog

```
:- use_module(library(clpfd)).

mm([S,E,N,D,M,O,R,Y], Type) :-
    domain([S,E,N,D,M,O,R,Y], 0, 9),
    S#>0, M#>0,
    all_different([S,E,N,D,M,O,R,Y]),
    sum(S,E,N,D,M,O,R,Y),
    labeling(Type, [S,E,N,D,M,O,R,Y]).
sum(S, E, N, D, M, O, R, Y) :-
    1000*S + 100*E + 10*N + D
    + 1000*M + 100*O + 10*R + E
    #= 10000*M + 1000*O + 100*N + 10*E +
Y.
```

```
| ?- mm([S,E,N,D,M,O,R,Y],[]).
D = 7,
E = 5,
M = 1,
N = 6,
O = 0,
R = 8,
S = 9,
Y = 2
```

Lambda Prolog

```
module  hc_interp.
accumulate lists.
type   backchain   (list form) -> form -> o.
type   try_clause  (list form) -> form -> form -> o.

hc_interp Cs (some B)  :- !, hc_interp Cs (B T).
hc_interp Cs (B and C) :- !, hc_interp Cs B, hc_interp Cs C.
hc_interp Cs (B or C) :- !, (hc_interp Cs B ; hc_interp Cs C).
hc_interp Cs A  :-  backchain Cs A.

backchain Cs A :- memb D Cs, try_clause Cs D A.

try_clause Cs (D1 and D2) A :-
     !, (try_clause Cs D1 A ; try_clause Cs D2 A).
try_clause Cs (all D) A :- !, try_clause Cs (D T) A.
try_clause Cs A A.
try_clause Cs (G imp A) A :- hc_interp Cs G.
```

Campos de Aplicación.

- Desarrollo de aplicaciones de inteligencia artificial.
- Construcción de sistemas expertos.
- Utilización de bases de datos de conocimiento.
- Procesamiento de lenguaje natural y la programación automática.
- Aplicaciones que implican búsqueda de patrones o información incompleta.

PROLOG.

PROLOG.

Prolog es un lenguaje de programación declarativo basado en la lógica de primer orden, particular-mente en una restricción de la forma clausal de la lógica

***PRO**grammation en **LOG**ique.*

Historia

- En 1981 apareció el primer intérprete de Prolog para ordenadores personales (en aquel tiempo, «microordenadores»), realizados con microprocesadores de ocho bits.
- En 1981, Prolog se adoptó como un elemento de partida para el proyecto japonés de «quinta generación», contribuyendo a la difusión del lenguaje.

Historia

- Un impulso renovador se produjo con la idea de ampliar su alcance mediante la introducción de construcciones para resolver problemas de satisfacción de restricciones (Jaffar y Lassez, 1987)
- Integración en las metodologías de orientación a objetos (Moura, 2004).

Características

Se especifican hechos y propiedades del problema; el sistema busca la solución usando:

- Hechos y reglas para representar la información.
- Deducciones para responder consultas.
- Verificación de las transformaciones, evaluadas partiendo de metas.
- La solución final resulta de aplicar resultados intermedios de las variables de la meta inicial.

PROLOG

- Prolog es un lenguaje usado para hacer computación simbólica y lógica
- Es declarativo, un programa se compone de una base de conocimientos que se compone de reglas y hechos que pueden generar nuevos hechos
- El intérprete de prolog da respuestas a solicitudes recorriendo los hechos y reglas de la base de conocimiento

PROLOG

Sintaxis de **PROLOG**

- Términos.
- Predicados.
- Hechos y Reglas.
- Queries.(preguntas)

Términos en PROLOG

- **Identificadores:** secuencias de letras minúsculas o dígitos y el guión bajo
  - ej: maria,ana,x25,x_25,alpha_omega
- **Números**: 1.221,2,3.03
- **Cadenas de caracteres:** secuencias de caracteres entre comillas simples
  - ej: 'maria','1.01','Cadena'
- **Variables:** secuencias de caracteres que comienzan con mayúsculas o guión bajo
  - ej:_x,Ana, Maria.

Listas en PROLOG

Son términos estructurados representados de una forma especial.

- [a,b,c,d]
- cada lista tiene [head|rest]
  - head=a rest[b,c,d]
- los términos de la lista pueden ser cualquier término
  - [a, f(a), 2, 3+5, point(X,1.5,Z)]

Predicados en PROLOG

- predicados: <identificador>(Term1, ..., Termk)
  - elefante(mary)
  - invita(john,megan)
- hechos:un hecho es un predicado terminado con punto('.').
  - <identifier>(Term1, ..., Termk).
- un hecho es una afirmación
  - elefante(mary). %maria es un elefante.

Reglas en PROLOG.

reglas:Sentencias condicionales de la forma

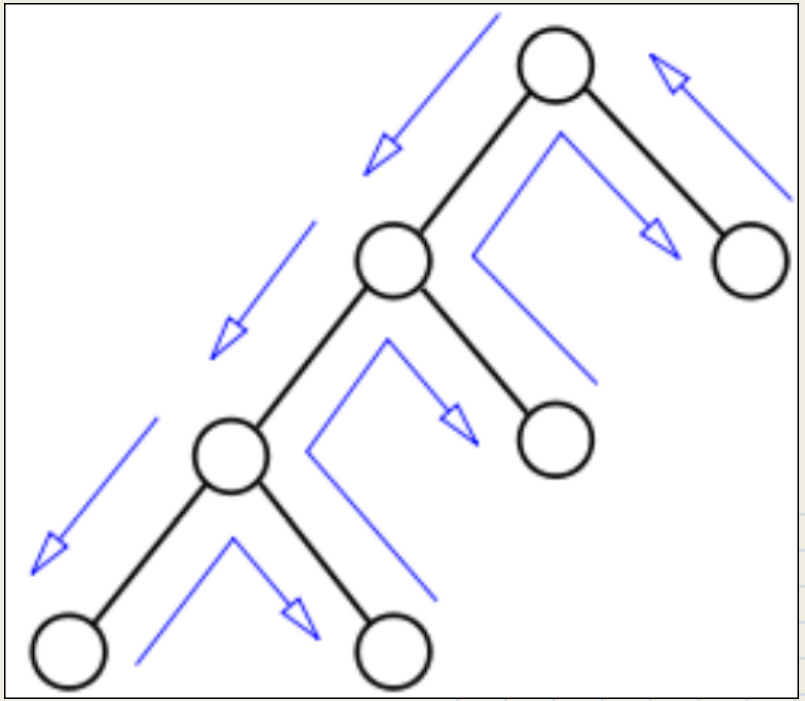
predicateH :- predicate1, ..., predicatek,

abuelo(X,Y):- progenitor(X ,Z), progenitor(Z,Y).
tio(X,Y):- progenitor(Z,Y), progenitor(V,Z), progenitor(V,X).

Reglas codifican formas de derivar o calcular un hecho nuevo

Backtraking

Prolog utiliza un sistema de backtracking para resolver una meta propuesta. El procedimiento de backtracking consiste en generar un árbol de búsqueda de todas las posibles resoluciones que puede tener la meta en función de la base de conocimiento.



Algunos Predicados Básicos.

- Predicados (operadores) de comparación
  - X < Y, X > Y, X <= Y, X >= Y, X = Y, X \ = Y, X == Y, X == Y, X\ == Y.
- Predicados de comprobación de tipos
  - atom(X),compound(X),float(X),integer(X),nonvar(X),rational(X),real(X), string(X),var(X)
- Entrada/Salida estándar
  - currentop(?Precedence,?Type,?Name),display(+Term),flush,get(-Char),get0(-Char),nl,op(+Precedence,+Type, +Name),put(+Char),read(-Term),skip(+Char),tab(+Amount),write(+Term),writeq(+Term),
- Funciones para la Depuración de Programas
  - debug,debugging,nodebug,nospy(+Pred),nospyall,notrace,spy(+Pred), trace,tracing.

PROLOG - EJEMPLOS

hoja1.pl [modified]

File Edit Browse Compile Prolog Pce Help

hoja1.pl [modified]

```
%    ?- carne(X).
%    ?- carne(X), postre(X).
%     ... etc...
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

entrada(paella).
entrada(gazpacho).
entrada(consome).
carne(filete_de_cerdo).
carne(pollo_asado).
pescado(trucha).
pescado(bacalao).
postre(flan).
postre(nueces_con_miel).
postre(naranja).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Definir la relación "plato_principal(X)" que
% indicara que un plato principal es un plato de carne o de
% pescado.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

plato_principal(X):-
        carne(X).
plato_principal(X):-
        pescado(X).
```

SWI-Prolog (AMD64, Multi-threaded, version 7.2.0)

File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 7.2.0)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- plato_principal(pollo_asado).
true .

2 ?- plato_principal(flan).
false.
```

PROLOG - EJEMPLOS

hoja1.pl

File Edit Browse Compile Prolog Pce Help

hoja1.pl | hoja2.pl | hoja3.pl | hoja4.pl [modified]

```
padre(andres,bernardo).
padre(andres,belen).
padre(andres,baltasar).
padre(baltasar,david).
padre(david,emilio).
padre(emilio,francisco).
madre(ana,bernardo).
madre(ana,belen).
madre(ana,baltasar).
madre(belen,carlos).
madre(belen,carmen).
abuelo(X, Y):-
        padre(X, Z),
        padre(Z, Y).
abuelo(X, Y):-
        padre(X, Z),
        madre(Z, Y).
progenitor(X, Y):-
        padre(X, Y).
progenitor(X, Y):-
        madre(X, Y).
mujer(belen).
mujer(ana).
mujer(carmen).
nieta(X, Y):-
        mujer(X),
        progenitor(Y, Z),
        progenitor(Z, X).
antepasado(X, Y):-
        progenitor(X, Y).
antepasado(X, Y):-
        progenitor(X, Z),
        antepasado(Z, Y).
descendiente(X, Y):-
        antepasado(Y, X).
```

SWI-Prolog (AMD64, Multi-threaded, version 7.2.0)

File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 7.2.0)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- antepasado(andres,francisco).
true .

2 ?- nieta(ana,carmen).
false.

3 ?- nieta(carmen,ana).
true
```

Referencias

- https://carloscatzin.wordpress.com/2010/03/29/inteligencia-artificial-programacion-logica-por-encima-de-la-funcional/
- http://www.di-mare.com/adolfo/cursos/2007-2/pp-Prolog.pdf
- http://artemisa.unicauca.edu.co/~ahurtado/matematicas/Funcional.pdf
- web.ing.puc.cl/~marenas/iic2213-11/clases/lpo.pdf
- www.uv.mx/aguerra/documents/2011-mpi-02.pdf
- gpd.sip.ucm.es/jaime/pl/sld.pdf
- www.sc.ehu.es/jiwnagom/FLP/FLP-archivos/LengLogicos.pdf
- www.mercurylang.org/documentation/papers/book.pdf
- https://sites.google.com/site/lambdaprologvideotutorial/

Referencias

- mural.uv.es/mijuanlo/PracticasPROLOG.pdf
- www.cs.toronto.edu/~hojjat/384w10/PrologTutorial1.pdf
- www.itnuevolaredo.edu.mx/takeyas/Apuntes/.../Apuntes/IA/Prolog.pdf
- arantxa.ii.uam.es/~dcamacho/logica/material/predicados.pdf
- www.gedlc.ulpgc.es/docencia/lp/documentacion/GB_Prolog.pdf
- platea.pntic.mec.es/jdelucas/prolog.htm
- http://teyjus.cs.umn.edu/old/examples/handbook-logic/hc_interp.mod

Gracias.