Programación Logica

Teórico

Andrés Felipe López Gutiérrez
Diego Alejandro Carvajal Beltran
Harold Alfredo Díaz Ortiz
Juan Felipe Callejas Fracica

Contenidos

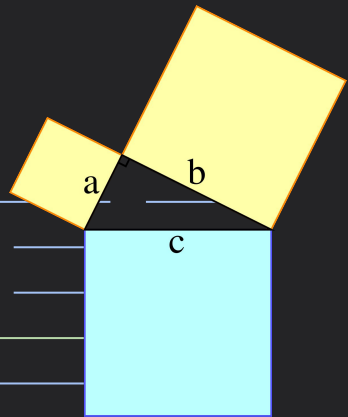
01

Filosofía del paradigma

Orígenes

Frase:
A mediados de los 70’tas alguien quería dejarle el trabajo de probar teoremas a las computadoras.

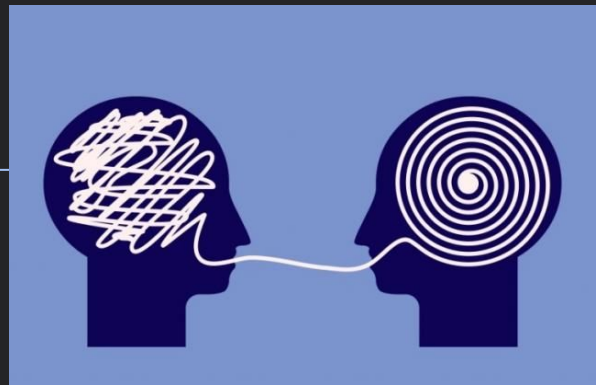
Explicación:
La programación lógica surge debido al interés por crear pruebas automatizadas de teoremas. Es decir, un sistema capaz de deducir.
Lo cual golpea el desarrollo de la inteligencia artificial.



Filosofía

Frase:
La filosofía de la programación lógica, es la filosofía lógica.

Explicación:
A partir de la filosofía lógica construimos una serie de objetos matemáticos que nos permiten llegar a preguntas formales que una computadora puede entender



Filosofía

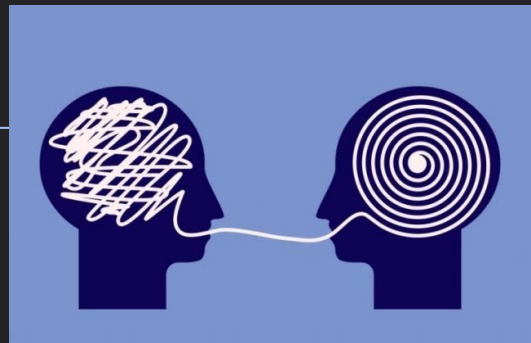
Naturaleza de la lógica

Frase:
La lógica es razón, discurso, o lenguaje.

Explicación:

- Es razón porque estudia la relación entre premisas y conclusiones, conectando cada tema nuevo con verdades ya interiorizadas.

- Es discurso y lenguaje porque nos sirve para explicar y para difundir el comportamiento de nuestro entorno.



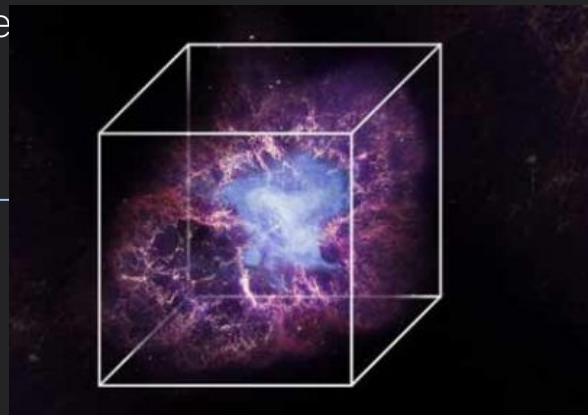
Filosofía

Naturaleza de la lógica

Frase:
La lógica es decidir si una interpretación es correcta

Explicación:

- Una inferencia es válida si la conclusión se desprende de las premisas, es decir, si la verdad de las premisas ase la verdad de la conclusión.



02

Conceptos claves

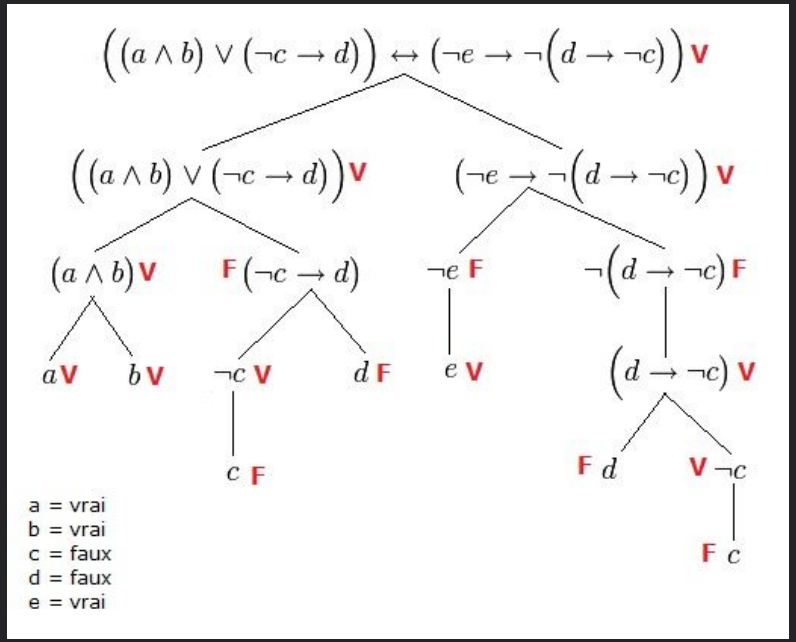
Conceptos

Concepto:

Lógica proposicional

Explicación:

- Representa un lenguaje formal mediante proposiciones vinculadas entre sí por conectores lógicos. El resultado de estos enunciados declarativos puede ser verdadero o falso.



01

Conceptos

Concepto:

Lógica de primer orden

Explicación:

- Extiende la lógica proposicional añadiendo predicados, representados mediante hechos, es decir, expresiones atómicas formuladas de la forma P(t1,...,tn).

English	First-Order
At least one x is P	$\exists x P(x)$
All x are P	$\forall x P(x)$
Some x are P	$\exists x P(x)$
Not all x are P	$\exists x \neg P(x)$
No x are P	$\forall x \neg P(x)$

Conceptos

Concepto:

Hechos y aridad

Explicación:

- Los hechos son declaraciones, cláusulas o proposiciones en las que se verifica un predicado P sobre los objetos t1,...,tn. El número de dichos objetos en el hecho se denomina aridad.

Arity	Examples
Nullary	5, False, constants
Unary	P(x), $\neg x$
Binary	x $\vee$ $y, x \wedge y$
Ternary	if p then q else r, $(p \rightarrow q) \wedge (\neg p \rightarrow r)$

Conceptos

Concepto:

Cláusulas y reglas

$$a \vee b \vee c$$

$$(\forall U)(\ healthy(U) \vee \neg eat(U, porridge)\ )$$

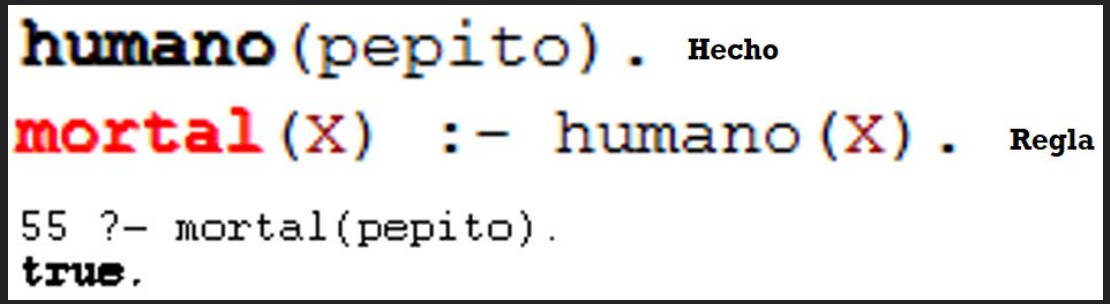
Explicación:

- Expresiones formadas por un conjunto finito de literales. Si una cláusula está formada con, máximo, un literal positivo, se conoce como cláusula de Horn. Un conjunto de cláusulas de Horn se conoce como reglas, de las que se infiere el valor de verdad de nuevas proposiciones.

Conceptos

Concepto:

Consultas



Explicación:

- Proposiciones construidas para que se deduzca su valor de verdad, partiendo de unos hechos y siguiendo unas reglas o cláusulas a manera de silogismos.

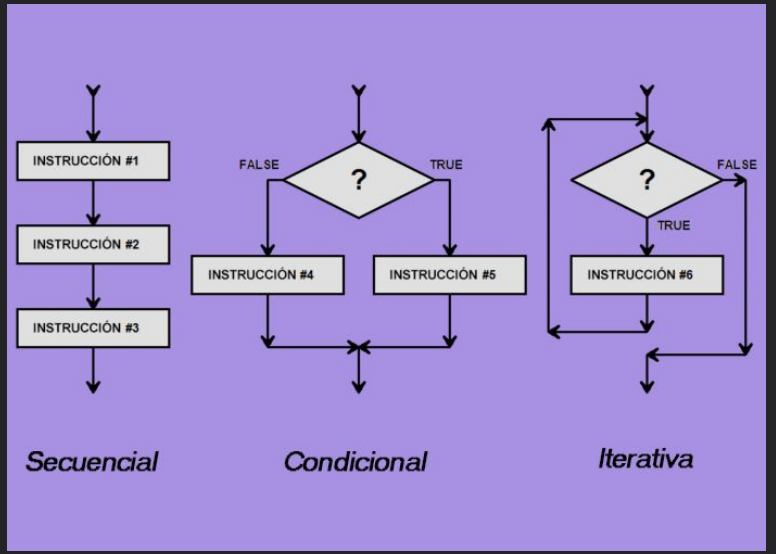
Conceptos

Concepto:

Estructuras de control

Explicación:

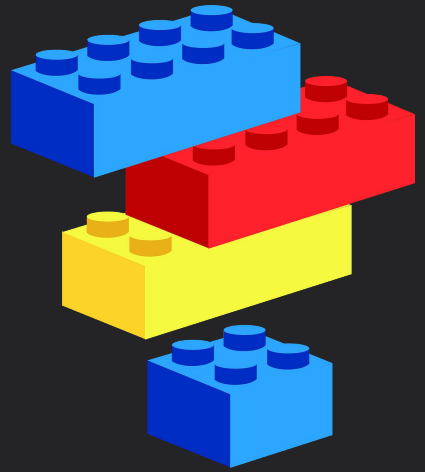
- Determinan las estrategias de demostración, su alternancia y orden que incide en cómo se ejecuta un programa lógico. La unión entre los componentes lógicos y estas estructuras determina el algoritmo básico del programa.



Conceptos

Concepto:

Modularidad



Explicación:

- Permite entender un programa mediante su fragmentación en problemas más pequeños y hallar la solución de estos para luego unirlos.

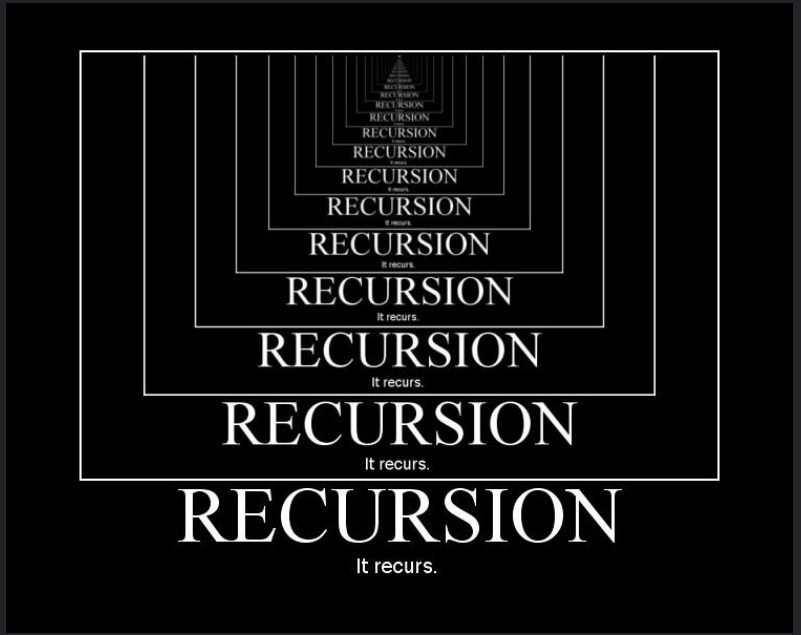
Conceptos

Concepto:

Recursión

Explicación:

- Es la forma de especificar procesos bajo sus propios términos de definición. Un problema puede resolverse definiendo el valor de problemas modulares llegando hasta un caso base.



03 Ventajas y desventajas

De la programación Lógica

Ventajas

Se puede aplicar Optimización.

Explicación:

- Se puede mejorar la eficiencia del algoritmo sin modificar su lógica, solo modificando los componentes.

01

Ventajas

Reconocimiento del lenguaje natural.

Explicación:

- Un programa es capaz de comprender la información contenida en una expresión lingüística humana, esto lo logra con ciertas limitaciones pero de forma efectiva.

Ventajas

Base de desarrollo.

Explicación:

- Al ser un paradigma casi indispensable en el desarrollo, se puede usar para apoyar y fortalecer otros paradigmas.

Ventajas

Facilidad de leer y comprender.

Explicación:

- Al ser basado en un sistema matemático casi cualquier persona con conocimientos básicos en estos puede leer un algoritmo y entenderlo.

Desventajas

Pocas áreas de aplicación.

Explicación:

- Su función principalmente era para problemas muy específicos, esto evita que se pueda aplicar a algunas áreas complejas o más amplias.

01

Desventajas

Codigos extensos.

Explicación:

- Aunque es muy optimizable eso no evita que los algoritmos que se creen lleguen a ser muy grandes para realizar una tarea específica y la misma optimización sea complicada de aplicar.

Desventajas

Hay mejores opciones.

Explicación:

- Se quedan bastante atrás en términos de interfaz, depuración y portabilidad.
- No existe una forma adecuada de representar los conceptos computacionales que se encuentran en mecanismos de variables de estado.

Desventajas

Poca comunidad.

Explicación:

- Al no ser muy popular la comunidad no invierte mucho en actualizar y mejorar los lenguajes y demás herramientas.

Lenguajes de programación

04

Prolog

- Es un lenguaje de programación enmarcado en los paradigmas de programación lógico y declarativa.
- Surge en la década de los 70 en Francia.
- Los programas en Prolog se componen de cláusulas de Horn.
- La ejecución de los programas se basan en los conceptos de *unificación* y *backtracking*

Mercury

- Mercury es un lenguaje de programación lógico/funcional y de propósito general.
- Combina la claridad y la expresividad de la programación declarativa con análisis estático avanzado y características de detección de errores.
- La sintaxis de Mercury está basada en la sintaxis de Prolog.

Godel

- Es un lenguaje declarativo, de propósito general y pertenece a los lenguajes lógicos.
- Lenguaje fuertemente tipado, con un sistema de tipado basado en múltiples órdenes y polimorfismo.
- Tiene facilidades centradas en la meta-lógica, facilitando la metaprogramación.
- Nombrado así en honor al lógico Kurt Godel

Datalog

- Lenguaje lógico y declarativo en el cual cada fórmula es una cláusula de Horn.
- Es un sistema deductivo de bases de datos en el que se escribe en lenguaje lógico.
- Sintaxis basada en Prolog.
- Generalmente utiliza un modelo de evaluación tipo bottom-up (de abajo hacia arriba)

05 Ejemplos

Ejemplo en Prolog

Vínculo

https://swish.swi-prolog.org/p/Negation%20as%20failure.swinb

Ejemplo en Mercury

```
:- module rot13.
:- interface.
:- import_module io.
:- pred main(io::di, io::uo) is det.
:- implementation.
:- import_module char,int,require.

main(!IO) :-
    io.read_char(Result, !IO),
    ( if Result = ok(Char) then
        rot13(Char, Rot13Char),
        io.write_char(Rot13Char, !IO),
        main(!IO)
    else if Result = eof then
        true
    else
        error("read failed")
    ).
```

```
:- pred rot13(char::in, char::out) is det.
rot13(Char, Rot13Char) :-
    char.to_int(Char, Code),
    ( if 0'A =< Code, Code =< 0'Z then
        Rot13Code = (Code - 0'A + 13) mod 26 + 0'A
    else if 0'a =< Code, Code =< 0'z then
        Rot13Code = (Code - 0'a + 13) mod 26 + 0'a
    else
        Rot13Code = Code
    ),
    ( if char.to_int(Ch, Rot13Code) then
        Rot13Char = Ch
    else
        error("Error")
    ).
```

Ejemplo en Datalog

```
from pyDatalog import pyDatalog
from pyDatalog.pyDatalog import create_terms as terms
from pyDatalog.pyDatalog import ask

pyDatalog.create_terms('scale')
terms('A,B,C,V')

scale['meter','inch']= 39.3700787
scale['mile','inch'] = 63360.0
scale['feet','inch'] = 12.0
scale['meter','cm'] = 100.0
scale['Km','meter'] = 1000.0

scale[A,B] = 1/scale[B,A]

print("Escalas")
print("Pulgada a metros:", scale['inch','meter'] == V, "\n")
print("Milla a metros:",scale['mile','meter'] == V, "\n")
print("Pies a centimetros:",scale['feet','cm'] == V, "\n")
print("Kilometro a milla:",scale['Km','mile'] == V , "\n")

terms('conv')
print("Conversion")
conv[V,A,B] = V * scale[A,B]
print("3 millas a metros:",conv[3,'mile','meter'] == V, "\n")
print("2.7 pulgadas a centimetros:",conv[2.7,'inch','cm'] == V, "\n")
print("0.74 millas a Kilometros:",conv[0.74,'mile','Km'] == V, "\n")
```

Ejemplo en Datalog

```
Escalas
Pulgada a metros: V
-----------------
0.025400000025908

Milla a metros: V
-----------------
1609.344001641531

Pies a centimetros: V
------------------
30.480000031089602

Kilometro a milla: V
------------------
0.6213711916035353
```

```
Conversion
3 millas a metros: V
------------------
4828.032004924593

2.7 pulgadas a centimetros: V
-------------------
6.8580000069951605

0.74 millas a Kilometros: V
-----------------
1.190914561214733
```

06

Aplicaciones del paradigma

Aplicaciones

01
Inteligencia artificial

02
Bases de datos

03
Reconocimiento de LN

04
Sistemas expertos

Reconocimiento de Lenguaje Natural

Primera aplicación

- Alain Colmerauer y Philippe Roussel - Universidad de Marsella (Francia - 1973)
- Fue la primera aplicación de programación lógica.
- Base para el desarrollo de ProLog

Gramáticas de metamorfosis

- Permitió reescribir símbolos terminales y no terminales.

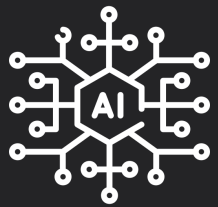
Sistema Experto

Área de conocimiento

Base de conocimiento (cuerpo)

Motor de inferencia

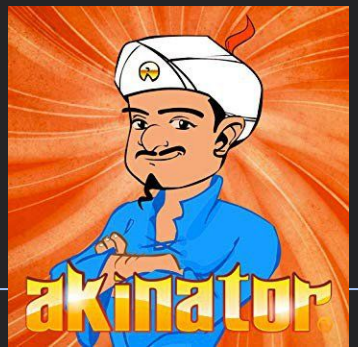
Puede verse como una IA



Retroalimentación

Teorema - colorario

Akinator



Ejemplos de programas

Genexus y Linx (Plataformas Low-code)
Plataformas tipo Low-Code para desarrollo de software.
Estas han sido desarrolladas utilizando Prolog y los principios de la programación lógica.

Aplicación en otro paradigma

Programación lógica probabilística

La programación probabilística es un paradigma de programación en el que se especifican modelos probabilísticos y la inferencia a partir de estos modelos se realizan automáticamente.

Con la programación lógica probabilística se seguirá el mismo funcionamiento que en la programación lógica, pero teniendo en cuenta que los hechos presentados no tienen por qué ocurrir siempre, como ocurre en la vida real.

Problog

Problog es un lenguaje de programación probabilístico que se basa en Prolog.

Referencias

- *Prolog. Wikipedia. Tomado de: https://es.wikipedia.org/wiki/Prolog*
- *Mercury. Mercurylang. Tomado de: https://mercurylang.org/about.html*
- *The Gödel Programming Language. Tomado de: https://dennisdarland.com/my_sw_projects/goedel/v1_3_27/doc/book.pdf*
- *McCarthy, J. (n.d.). Datalog: Deductive Database Programming. Tomado de: https://docs.racket-lang.org/datalog/*
- *Datalog. Wikipedia. Tomado de: https://en.wikipedia.org/wiki/Datalog*
- *Ramírez, L. ¿Qué es una plataforma low code o programación de bajo código? IEBS. Tomado de: https://www.iebschool.com/blog/que-es-low-code-big-data/*
- *Genexus.Tomado de: https://www.genexus.com/es/*
- *Programación lógica. Paradigmas de programación. Tomado de : http://ferestrepoca.github.io/paradigmas-de-programacion/proglogica/logica_teoria/index.html*
- *https://www.doc.ic.ac.uk/~cclw05/topics1/summary.html*
- *https://www.baeldung.com/cs/first-order-logic*

Gracias!

¿Preguntas?