Programación lógica

Angel David Corredor
Nicolás Gómez Gutiérrez
María Alejandra Robayo

Lenguajes de Programación
Universidad Nacional de Colombia
Presentada el: 19 de Junio de 2019

Contenido

- ✓ Clasificación de lenguajes de programación
- ✓ Filosofía del paradigma
- ✓ Conceptos claves
- ✓ Ventajas y desventajas
- ✓ Lenguajes de programación lógica
- ✓ Ejemplos en distintos lenguajes
- ✓ Aplicaciones de este paradigma
- ✓ Referencias/Bibliografía

$$\begin{array}{l} p \\ \underline{p \to q} \\ \therefore q \end{array}$$

Clasificación

Lenguajes de programación

- **Imperativos**
  - ¿Cómo llegar?
  - Declaración de algoritmos que solucionan el problema.
- **Declarativos**
  - ¿A dónde debo llegar?
  - Descripción del problema.
  - Utilización de inferencia.

Clasificación

Lenguajes de programación

- **Imperativos**
  - Orientados a Objetos
  - Por Procedimientos
  - Procesamiento En Paralelo
- **Declarativos**
  - Funcionales
  - Lógicos
  - Relacionales

Filosofía del paradigma

"Modelar problemas por medio de la abstracción, utilizando un sistema de lógica formal que permite llegar a una conclusión por medio de hechos y reglas".

Filosofía

- EsAve(pingüino)
- EsAve(paloma)
- EsAve(canario)
- EsAve(loro)

Filosofía del paradigma

No se busca un algoritmo que resuelva el problema, se proporcionan las bases para que el lenguaje de programación lógica lo resuelva a través de deducción controlada.

Filosofía

- EsAve(pingüino)
- EsAve(paloma)
- EsAve(canario)
- EsAve(loro)

EsAve(X) ⟶ Vuela(X)

Filosofía

- EsAve(pingüino)
- EsAve(paloma)
- EsAve(canario)
- EsAve(loro)

EsAve(X) ⟶ Vuela(X)

¿  ?

Filosofía del paradigma

¿Qué problema resuelve?

Dado un problema S, resuelve si la afirmación A es solución o no de S (o en qué casos lo es).

Filosofía

- EsAve(pingüino)
- EsAve(paloma)
- EsAve(canario)
- EsAve(loro)

$$EsAve(X) \longrightarrow Vuela(X)$$

$$EsAve(X) \land no(X, pingüino) \longrightarrow Vuela(X)$$

Filosofía del paradigma

"Un programa lógico consta de un conjunto de fórmulas lógicas que expresan propiedades satisfechas por un cierto problema."

Conceptos clave

- Lógica proposicional.
- Lógica de primer orden.
- Lógica de orden superior.

Lógica proposicional

Proposición $\longrightarrow$ Proposición Atómica | Proposición Compleja

Proposición atómica $\longrightarrow$ **Verdadero** | **Falso** | Símbolo Proposicional

Símbolo proposicional $\longrightarrow$ **P** | **Q** | **R** | …

Proposición compleja $\longrightarrow$ $\neg$ Proposición
| (Proposición $\wedge$ Proposición)
| (Proposición $\vee$ Proposición)
| (Proposición $\Rightarrow$ Proposición)
| (Proposición $\Leftrightarrow$ Proposición)

Lógica proposicional

Ejemplos:

$5 + 20$

Falso $\wedge$ ($P \vee Q$)

$(P \wedge Q) \Rightarrow \neg R$

S

Lógica proposicional

Ejemplos:

- ✗ $5 + 20$
- ✓ $\text{Falso} \land (P \lor Q)$
- ✓ $(P \land Q) \Rightarrow \neg R$
- ✓ S

Lógica proposicional

Ejemplos:

- ✗ $5 + 20$
- ✓ $Falso \wedge (P \vee Q)$ — Proposición compleja
- ✓ $(P \wedge Q) \Rightarrow \neg R$ — Proposición compleja
- ✓ S — Proposición

Lógica de primer orden

Extiende la lógica proposicional permitiendo además el uso de cuantificadores y declarar predicados sobre diferentes objetos.

Lógica de primer orden

Extiende la lógica proposicional permitiendo además el uso de cuantificadores y declarar predicados sobre diferentes objetos.

¿Cuantificadores?

Cuantificadores

Operador sobre un conjunto de individuos permitiendo construir proposiciones sobre conjuntos.

Símbolo	Nombre	Lectura
$\forall$	Cuantificador universal.	Para todo…
$\exists$	Cuantificador existencial.	Existe un…
$\exists!$	Cuantificador existencial único.	Existe exactamente un…

Lógica de primer orden

Extiende la lógica proposicional permitiendo además el uso de cuantificadores ($\forall$ / $\exists$ / $\exists!$) y declarar predicados sobre diferentes objetos.

¿Predicados?

Predicados

Funciones sobre objetos que se usan para expresar propiedades o relaciones entre éstos.

$$\rho : \; A \rightarrow B$$

$$a \rightarrow b = P(a)$$

Predicados

Funciones sobre objetos que se usan para expresar propiedades o relaciones entre éstos.

$$
\begin{aligned}
\rho:\ & A \rightarrow B \\
& a \rightarrow b = P(a)
\end{aligned}
$$

EsAve(x)

Predicados

Funciones sobre objetos que se usan para expresar propiedades o relaciones entre éstos.

$$
\begin{aligned}
\rho :\ & C \times D \rightarrow B \\
& c, d \quad \rightarrow b = P(c, d)
\end{aligned}
$$

Predicados

Funciones sobre objetos que se usan para expresar propiedades o relaciones entre éstos.

$$\rho: \; C \times D \rightarrow B$$
$$c, d \;\; \rightarrow b = P(c, d)$$

Padre(x, y)

Estudia(x, y)

Lógica de primer orden

Extiende la lógica proposicional permitiendo además el uso de cuantificadores ($\forall$ / $\exists$ / $\exists!$) y declarar predicados (propiedades) sobre diferentes objetos.

Lógica de primer orden

Ejemplos:

$\forall x\,(\,Ave(x) \land \neg no(x, pingüino) \Rightarrow Vuela(x)\,)$

$\exists x\,(\,Vuela(x) \land \neg Ave(x)\,)$

$\exists ! x\,(\,Ave(x) \land \neg Vuela(x)\,)$

$\forall y\ \exists ! x\,(\,Madre(x, y)\,)$

Lógica de orden superior

Extiende la lógica de primer orden, permitiendo reducir conjuntos (como podrían ser las proposiciones) a una variable sobre la cual se pueden expresar nuevas proposiciones o hacer uso de los cuantificadores.

$$\forall P \, \forall x \, (Px \lor \neg Px)$$

Cláusula de horn

Disyunción de literales con máximo un literal positivo.

$$\neg p \lor \neg q \lor \neg r \ldots \lor u$$

$$(p \land q \land r \ldots) \Rightarrow u$$

Conceptos clave

Consultas

Hechos

Reglas

Hecho

Expresión atómica que verifica una relación sobre un objeto.

Por ejemplo: El loro es un ave.

Reglas

Conjunto de proposiciones lógicas que permiten inferir el valor de verdad de una nueva proposición.

Por ejemplo: Todas las aves tienen alas. Todos los animales que tienen alas, ponen huevos.

Consultas

Proposición construida con el propósito de ser demostrada o de encontrar el conjunto de valores que la convierten verdadera.

Por ejemplo: ¿El loro pone huevos?

Aridad

Es un número que indica el número de variables individuales que utiliza el predicado para formar una oración.

Oración	Predicado	Aridad
Juan pasó.	pasó(Juan)	1
Juan pasó el parcial.	pasó(Juan, Parcial)	2
Juan pasó el parcial de cálculo.	pasó(Juan, Parcial, Cálculo)	3

Nombre según aridad

Aridad	Nombre del predicado
0	Enunciado
1	Propiedad
2 (o más)	Relación

Recursión

Especificación de un proceso basado en su propia definición.

Presenta 2 elementos clave:

- Caso base.
- Llamado recursivo.

Conceptos clave: Recursión

```
factorial(0, 1)
factorial(N, R) -> R = N * factorial(N-1)
```

Conceptos clave: Recursión

```
factorial(0, 1)
factorial(N, R) -> R = N * factorial(N-1)

mcd(N, 0, 0)
mcd(A, B, C) -> mcdAux(A, B, C)

mcdAux(N, 1, 1)
mcdAux(A, B, C) -> A < B ∧ mcd(B, A, C)
mcdAux(A, B, C) -> mcd(B, A%B, C)
```

Unificación

Proceso ejecutado sobre una variable para poder ser usada en la evaluación de una proposición.

Conceptos clave: Recursión

```
factorial(0, 1)
factorial(N, R) -> N1 = N−1 ∧ factorial(N1, R1) ∧ R = N * R1

mcd(N, 0, 0)
mcd(A, B, C) -> mcdAux(A, B, C)

mcdAux(N, 1, 1)
mcdAux(A, B, C) -> A < B ∧ mcd(B, A, C)
mcdAux(A, B, C) -> AB = A%B ∧ mcd(B, AB, C)
```

Motor de Inferencia

$$\begin{array}{l} \quad p \\ \quad \underline{p \rightarrow q} \\ \therefore q \end{array}$$

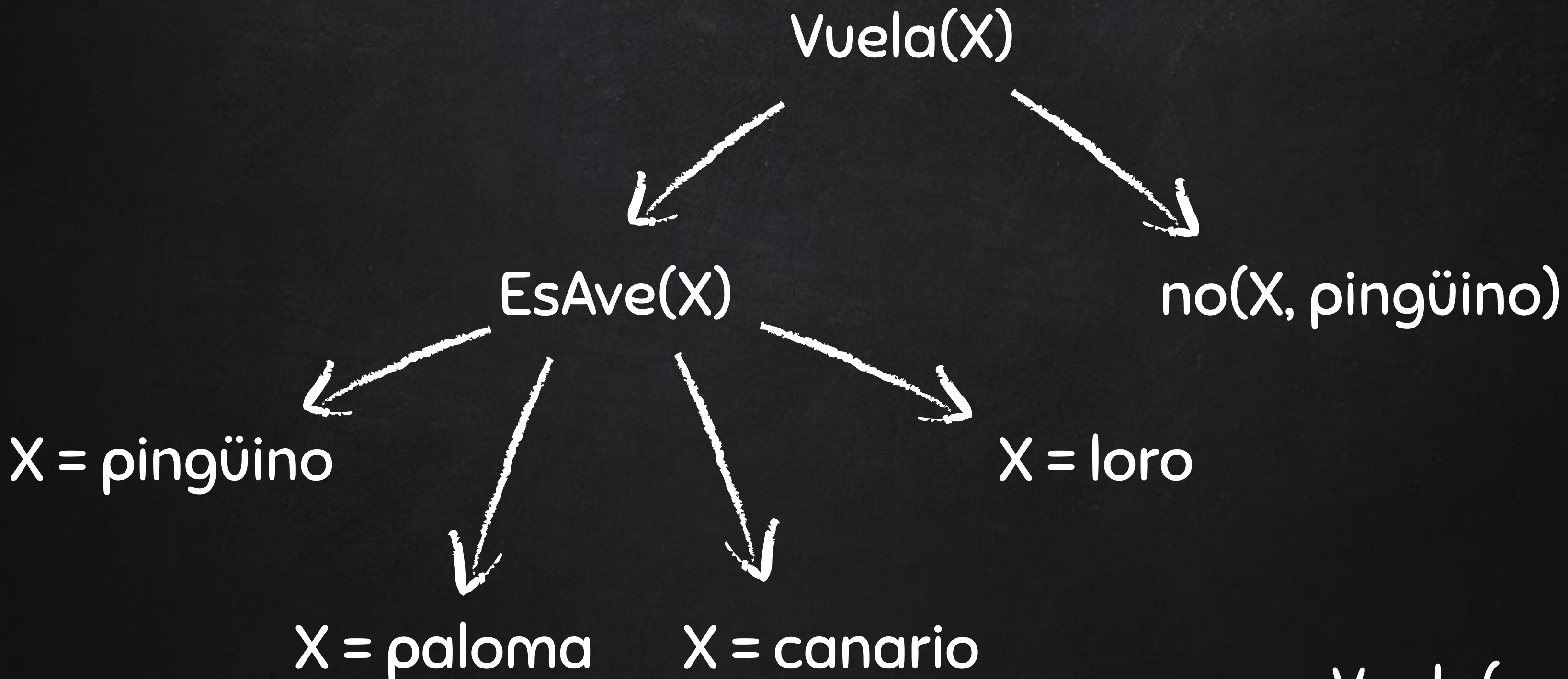
Conceptos clave

- EsAve(pingüino)
- EsAve(paloma)
- EsAve(canario)
- EsAve(loro)

$$\text{EsAve}(X) \longrightarrow \text{Vuela}(X)$$

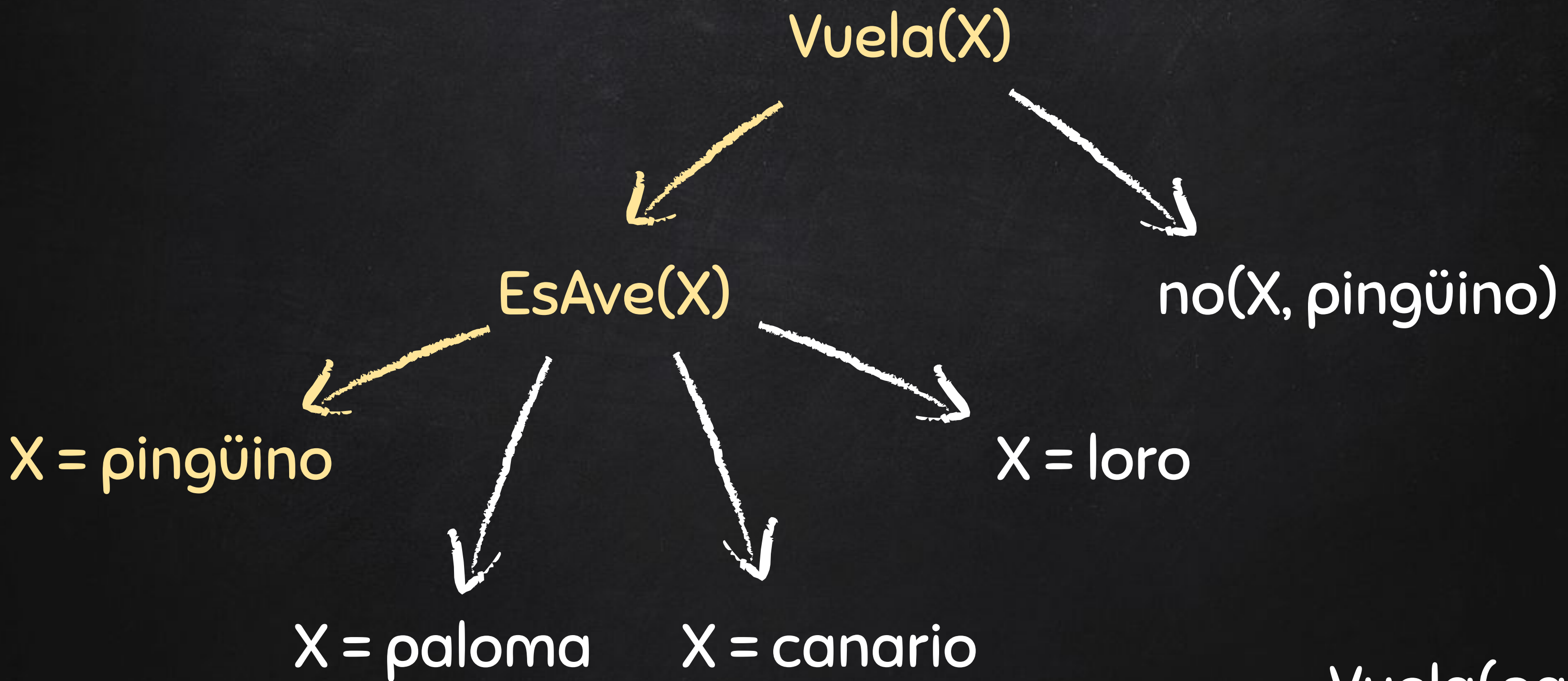
$$\text{EsAve}(X) \land \text{no}(X, \text{pingüino}) \longrightarrow \text{Vuela}(X)$$

Motor de Inferencia



Vuela(paloma)?

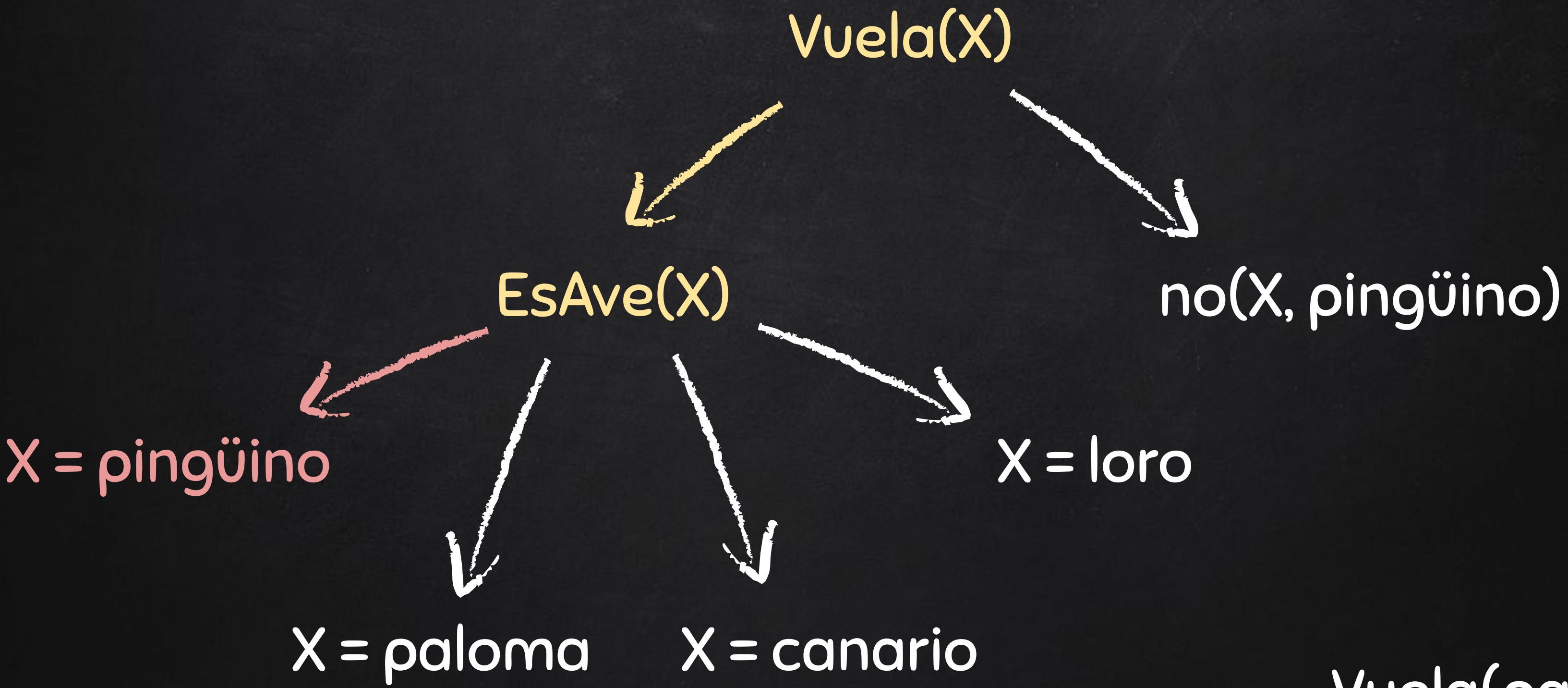
Motor de Inferencia



Vuela(paloma)?

Conceptos clave

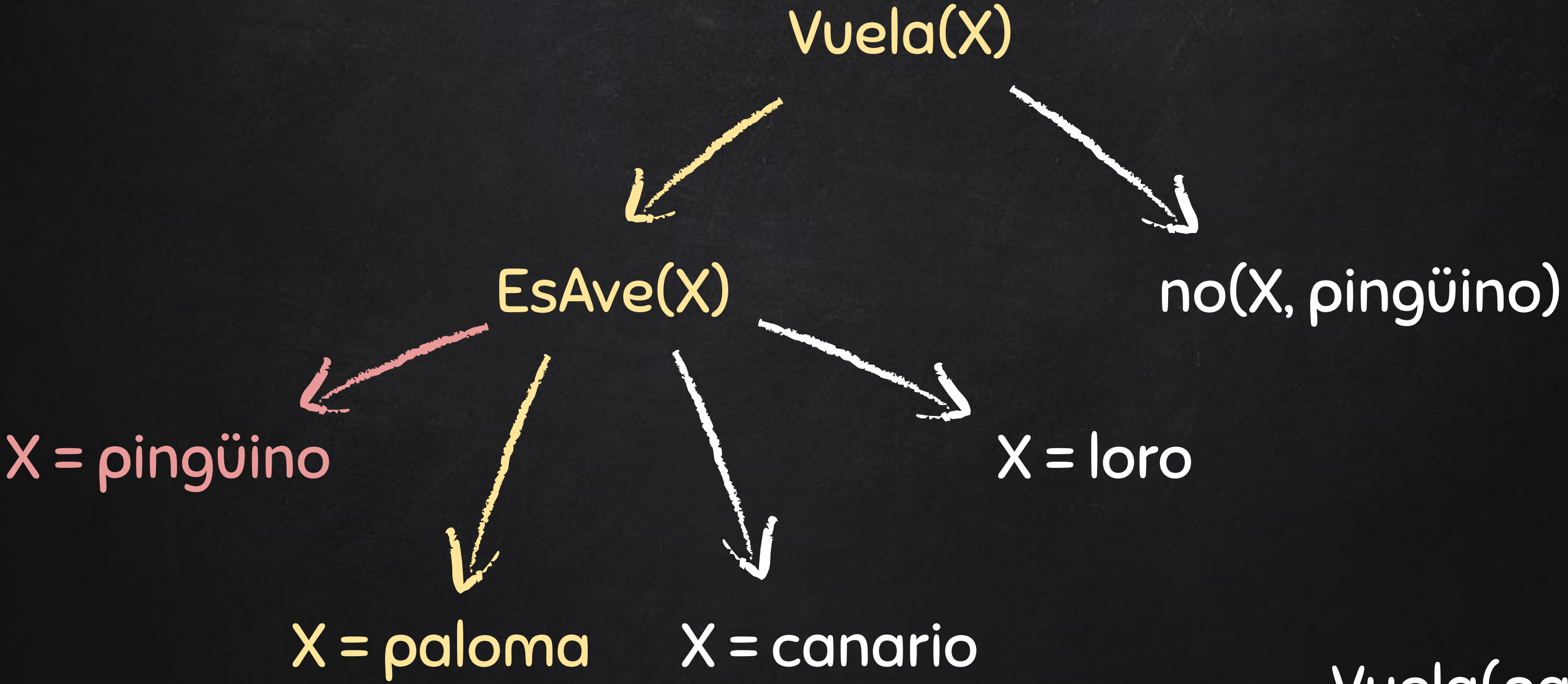
Motor de Inferencia



Vuela(paloma)?

Conceptos clave

Motor de Inferencia



Vuela(paloma)?

Conceptos clave

Motor de Inferencia

Vuela(X)

EsAve(X)

no(X, pingüino)

X = pingüino

X = paloma

X = canario

X = loro

Vuela(paloma)?

Conceptos clave

Motor de Inferencia

Vuela(X)

EsAve(X)

no(X, pingüino)

X = pingüino

X = loro

X = paloma

X = canario

Vuela(paloma)?

Conceptos clave

Motor de Inferencia

Vuela(X)

EsAve(X)

no(X, pingüino)

X = pingüino

X = paloma

X = canario

X = loro

Vuela(pingüino)?

Motor de Inferencia

Vuela(X)

EsAve(X)

no(X, pingüino)

X = pingüino

X = loro

X = paloma

X = canario

Vuela(gaviota)?

Ventajas y desventajas

Ventajas y desventajas

Abstracción del problema.
¿Ventaja o desventaja?

Ventajas y desventajas

✓ Descripciones independientes de la implementación (unificación semántica).

Ventajas y desventajas

- ✓ Descripciones independientes de la implementación (unificación semántica).
- ✓ Expresión simple y precisa de los problemas.

Ventajas y desventajas

- ✓ Descripciones independientes de la implementación (unificación semántica).
- ✓ Expresión simple y precisa de los problemas.

¿Y la resolución de ese problema?

Ventajas y desventajas

- ✓ Descripciones independientes de la implementación (unificación semántica).
- ✓ Expresión simple y precisa de los problemas.
- ✓ Puede llevar a una reducción de la complejidad.

Ventajas y desventajas

- ✓ Descripciones independientes de la implementación (unificación semántica).
- ✓ Expresión simple y precisa de los problemas.
- ✓ Puede llevar a una reducción de la complejidad.

- ✘ Puede llegar a ser extremadamente ineficiente.

Ventajas y desventajas

- ✓ Descripciones independientes de la implementación (unificación semántica).
- ✓ Expresión simple y precisa de los problemas.
- ✓ Puede llevar a una reducción de la complejidad.

- ✗ Puede llegar a ser extremadamente ineficiente.

¿Se puede optimizar la resolución de un problema?

Ventajas y desventajas

- ✓ Descripciones independientes de la implementación (unificación semántica).
- ✓ Expresión simple y precisa de los problemas.
- ✓ Puede llevar a una reducción de la complejidad.
- ✓ Permite su optimización sin modificar el código.

- ✘ Puede llegar a ser extremadamente ineficiente.

Ventajas y desventajas

- ✓ Descripciones independientes de la implementación (unificación semántica).
- ✓ Expresión simple y precisa de los problemas.
- ✓ Puede llevar a una reducción de la complejidad.
- ✓ Permite su optimización sin modificar el código.

- ✘ Puede llegar a ser extremadamente ineficiente.
- ✘ Dificultad en su depuración.

Ventajas y desventajas

- ✓ Descripciones independientes de la implementación (unificación semántica).
- ✓ Expresión simple y precisa de los problemas.
- ✓ Puede llevar a una reducción de la complejidad.
- ✓ Permite su optimización sin modificar el código.

- ✗ Puede llegar a ser extremadamente ineficiente.
- ✗ Dificultad en su depuración.
- ✗ Pocas herramientas disponibles.

Ventajas y desventajas

- ✓ Descripciones independientes de la implementación (unificación semántica).
- ✓ Expresión simple y precisa de los problemas.
- ✓ Puede llevar a una reducción de la complejidad.
- ✓ Permite su optimización sin modificar el código.

- ✗ Puede llegar a ser extremadamente ineficiente.
- ✗ Dificultad en su depuración.
- ✗ Pocas herramientas disponibles.

¿A dónde se va la base de conocimiento?

Ventajas y desventajas

- ✓ Descripciones independientes de la implementación (unificación semántica).
- ✓ Expresión simple y precisa de los problemas.
- ✓ Puede llevar a una reducción de la complejidad.
- ✓ Permite su optimización sin modificar el código.
- ✓ Base de conocimiento fácilmente escalable.

- ✘ Puede llegar a ser extremadamente ineficiente.
- ✘ Dificultad en su depuración.
- ✘ Pocas herramientas disponibles.

Ventajas y desventajas

- ✓ Descripciones independientes de la implementación (unificación semántica).
- ✓ Expresión simple y precisa de los problemas.
- ✓ Puede llevar a una reducción de la complejidad.
- ✓ Permite su optimización sin modificar el código.
- ✓ Base de conocimiento fácilmente escalable.

- ✗ Puede llegar a ser extremadamente ineficiente.
- ✗ Dificultad en su depuración.
- ✗ Pocas herramientas disponibles.
- ✗ Inferencia limitada por su base de conocimiento.

Ventajas y desventajas

✓ Descripciones independientes de la implementación (unificación semántica).
✓ Expresión simple y precisa de los problemas.
✓ Puede llevar a una reducción de la complejidad.
✓ Permite su optimización sin modificar el código.
✓ Base de conocimiento fácilmente escalable.

✗ Puede llegar a ser extremadamente ineficiente.
✗ Dificultad en su depuración.
✗ Pocas herramientas disponibles.
✗ Inferencia limitada por su base de conocimiento.

¿Y las aplicaciones?

Ventajas y desventajas

- ✓ Descripciones independientes de la implementación (unificación semántica).
- ✓ Expresión simple y precisa de los problemas.
- ✓ Puede llevar a una reducción de la complejidad.
- ✓ Permite su optimización sin modificar el código.
- ✓ Base de conocimiento fácilmente escalable.

- ✗ Puede llegar a ser extremadamente ineficiente.
- ✗ Dificultad en su depuración.
- ✗ Pocas herramientas disponibles.
- ✗ Inferencia limitada por su base de conocimiento.
- ✗ Áreas de aplicación muy específicas.

Lenguajes de programación lógica

- Prolog
- Gödel
  - polimorfismo
  - metaprogramación
  - altamente declarativo
- Datalog (python)

Ejemplo en Datalog

```
esAve( pingüino ).
esAve( paloma ).
esAve( canario ).
esAve( loro ).

vuela(x) :-  x \= pingüino , esAve(x).

?- vuela( paloma ).
```

Lenguajes de programación

Lenguajes que integran la programación funcional y lógica:

- λProlog
  - Derivado de Prolog
  - tipos polimórficos
  - módulos
  - tipos de datos abstractos
- Mercury
- Babel
- Escher (go)
- Curry
  - basado en Haskell

Ejemplo en curry

```
EsAve X | X == "pinguino" = True
      | X == "paloma" = True
      | X == "canario" = True
      | X == "loro" = True
      | otherwise = False

Vuela X | EsAve X && X /= "pinguino" = True
      | otherwise = False

main = Vuela "paloma"
```

Lenguajes de programación

Lenguajes que integran la programación orientada a objetos y lógica:

- Logtalk
  - extensión de prolog
- Visual Prolog
- Actor Prolog

Ejemplo en Prolog

```
esAve( pingüino ).
esAve( paloma ).
esAve( canario ).
esAve( loro ).

vuela(X) :-  esAve(X) , X \== pingüino .

?- vuela( paloma ).
```

Ejemplo en Prolog

```
separarmitad(L,L1,L2) :- separarmitadaux(L,[],[],L1,L2).
separarmitadaux([],L1,L2,L1,L2).
separarmitadaux([H|T],L1,L2,LL1,LL2) :- separarmitadaux(T,L2,[H|L1],LL1,LL2).

fusion([],L,L).
fusion(L,[],L).
fusion([H1|T1],[H2|T2],[H1|L]) :- H1 =< H2, fusion(T1,[H2|T2],L).
fusion([H1|T1],[H2|T2],[H2|L]) :- H1 > H2, fusion([H1|T1],T2,L).

ordenar([],[]).
ordenar([A],[A]).
ordenar(L,O) :- separarmitad(L,L1,L2), ordenar(L1,O1), ordenar(L2,O2), fusion(O1,O2,O).
```

Aplicaciones de este paradigma

- Sistemas Expertos
- Comprobación automática de teoremas
- Inteligencia Artificial
  - Sistemas basados en reglas
  - Reconocimiento de Lenguaje Natural
  - Búsqueda de patrones

Bibliografía/Referencias

- http://ferestrepoca.github.io/paradigmas-de-programacion/proglogica/logica_teoria/introduccion.html
- http://www.amzi.com/articles/code07_whitepaper.pdf
- https://upcommons.upc.edu/bitstream/handle/2117/93325/TJTM1de2.pdf
- http://diposit.ub.edu/dspace/bitstream/2445/64643/1/memoria.pdf