1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1

</ Clase 8 - Paradigma Lógico

- Características Programación Lógica.
- Base de conocimientos: Hechos y Reglas.
- Consultas.
- Motor de inferencias.
- Cómo programar en el paradigma.
- Usos y aplicación.
- Lenguaje PROLOG.
- Ejercicio para resolver

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</>

El Paradigma Lógico

} /> [

1011 011 01 1011001 10 11011 011 01 110110 110111 1101

</Características de la P. Lógica

Un programa, en este paradigma, es un conjunto de definiciones de funciones o ecuaciones matemáticas, cuya evaluación permite obtener el resultado de una consulta.

En el paradigma lógico se trabaja en forma descriptiva: el programa NO indica cómo resolver el problema sino establece relaciones entre las entidades del mismo, las cuales describen qué hacer. Por esta razón se lo encuadra en la programación declarativa.

</Características de la P. Lógica

Un programa lógico NO tiene un algoritmo que indique los pasos a seguir para llegar al resultado, sino que está formado por expresiones lógicas que declaran o describen la solución y es el sistema interno quien proporciona la secuencia de control en que se utilizan esas expresiones.

Tiene como característica principal el uso de reglas lógicas o predicados de 1º orden para inferir o derivar conclusiones a partir de datos.

En este paradigma, un algoritmo lógico se construye especificando una base de conocimiento (Hechos y reglas) sobre la que se realizan consultas.

</Base de Conocimientos

Está formada por hechos y reglas

Los hechos representan enunciados o predicados siempre verdaderos que muestran relaciones entre los datos del problema. (proposiciones).

Las reglas muestran relaciones entre los datos, los hechos y otras reglas, y permiten deducir o demostrar objetivos.

</Base de Conocimientos:Ejemplo

```
% clausulas [% comenta la línea]

hermano["juan", "pedro"].

hijo["juancito", "juan"].

tio[X, Y] :- hermano[Z, X], hijo[Y, Z].
```

Hechos

Regla

Implicación: de derecha a izquierda [sí → entonces]

</Consultas

Una consulta es un objetivo a ser demostrado por comparación o pattern matching contra la base de conocimiento.

Se trata de comparar la consulta con algún hecho o con la cabeza de una regla.

La consulta puede ser de dos tipos:

- Validación (cuya respuesta es Si o No)
- Búsqueda (que retorna los valores que hacen verdadero al objetivo).

</Consultas

Si puede demostrar la consulta u objetivo, responde:

- Yes (si es una validación)
- El/los valor/es que satisface/n la consulta (en caso de búsqueda)

Si no puede demostrar el objetivo, responde No, que tiene dos significados:

- Es falso
- No puede deducirlo con la información contenida en la base de conocimiento

</Consultas: Ejemplo

Base de Conocimiento

titulo(“divergente”).
titulo(“octubre un crimen”).
genero(“ciencia ficción”).
genero(“policial”).
generoLibro(“divergente”, “ciencia cicción”).

Consulta 1
titulo(“divergente”).
true.

Consulta 2
titulo(“insurgente”).
false.

Consulta 3
titulo(T).
T=”divergente”;
T=”octubre un crimen”

</Consultas: Variables

Si la consulta posee variables, las mismas se asocian o ligan con los valores de los hechos o con los argumentos de las cabezas de las reglas

```
Base de Conocimiento:
titulo(“divergente”).
titulo(“octubre un crimen”).
genero(“ciencia ficción”).
genero(“policial”).
generoLibro(“divergente”, “ciencia cicción”).
esDelGenero(TIT,GEN):-titulo(TIT),generoLibro(TIT,GEN)
.
```

</Consultas: Variables

Base de Conocimiento:
titulo(“divergente”).
titulo(“octubre un crimen”).
genero(“ciencia ficción”).
genero(“policial”).
generoLibro(“divergente”, “ciencia cicción”).
esDelGenero(TIT,GEN):-titulo(TIT),generoLibro(TIT,GEN).

Consulta 1
titulo(TIT)

Consulta 2
esDelGenero(“divergente”,GEN).

</Consultas: Variables

Consulta 2
esDelGenero(“divergente”,G).

¿Qué pasa en este caso?

esDelGenero(TIT,GEN):-titulo(TIT),generoLibro(TIT,GEN).

Se compara la consulta con la cabeza de la regla: liga “divergente” -> TIT y G -> GEN, reemplaza en el cuerpo de la regla y la aplica. titulo(“divergente”),generoLibro(“divergente”,G) y ahora tiene 2 subobjetivos nuevos a demostrar.

</Consultas: Variables

Se pueden hacer 3 tipos de consultas:
Sin variables
esDelGenero(“divergente”, “ciencia ficcion”) Validación

Con variables
esDelGenero(“divergente”, GEN) Búsqueda (de qué genero es divergente?)

Con variable anónima: para saber si existe algún objeto que haga verdadera la consulta
esDelGenero(_,”policial”) Búsqueda, y respuesta Si o No (existe algún libro de tipo policial).

</Cómo programar en el paradigma

A) Modelar el problema mediante una Base de Conocimiento formada por hechos que muestren las relaciones de los datos de dicho problema.

B) Formular luego reglas que se cumplan en el mundo del problema modelado.

C) Formular consultas al sistema para que demuestre o deduzca la respuesta a partir de la base de conocimiento usando resolución.

</Aplicaciones

Inteligencia Artificial
Sistemas expertos
Robótica
Procesamiento de Lenguaje Natural
Compiladores
Bases de datos deductivas
Acceso a bases de datos desde páginas web

</>

Sintaxis Prolog

} /> [

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Prolog: Sintaxis

Constantes y predicados comienzan con minúscula
Variables van mayúscula
Hechos y predicados terminan en punto
Se usa ; en una consulta para ver más resultados
El <enter> o el . finaliza la consulta
Los comentarios de líneas se realizan con %

</Prolog: Sintaxis

Además:
write[X] imprime el valor de X
read[X] ingresa un valor a X
== compara si son iguales
\== compara si son distintos
Operadores: + - / *
X mod Y
Operador is sirve para asignar valor [Ejem: Z is X + Y]

</Prolog: Sintaxis

Ejemplos de Reglas:

```
mayor[X,Y,Z]:- X>Y, Z is X.
mayor[X,Y,Z]:- X<Y, Z is Y.
mayor[X,Y,Y]:- X==Y, Z is 0.
```

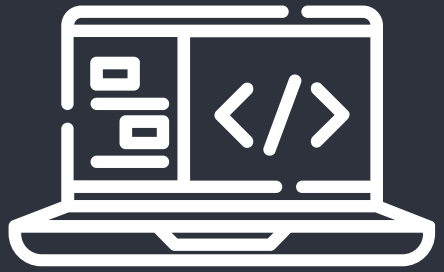
Otra forma de hacerlo:

```
mayor[X,Y,X]:- X>Y.
mayor[X,Y,Y]:- X<Y.
mayor[X,Y,0]:- X==Y.
```

```
posit[X]:- X>0. %se usa la hipótesis de mundo cerrado
```

</>

Ejercicio



} /> [

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Ejercicio: Base de Conocimientos

```
marca(fiat).
marca(toyota).
marca(renault).
marca(bmw).

modelo(fiat,bravo,2018).
modelo(fiat,600,1995).
modelo(toyota,corolla,2011).
modelo(toyota,hilux,1998).
modelo(toyota,etios,2017).
modelo(renault,clio,1999).
modelo(bmw,m3,1995).

kilometraje(bravo,0).
kilometraje(600,195000).
kilometraje(hilux,254250).
kilometraje(etios,15000).
kilometraje(corolla,100000).
kilometraje(clio,210000).
kilometraje(m3,45000).

precio(bravo,603500).
precio(600,56000).
precio(hilux,230000).
precio(corolla,450000).
precio(etios,367500).
precio(clio,21000).
precio(m3,95000).
```

</Ejercicio: Base de Conocimientos

1- Generar una regla que me permita mostrar el precio de los autos de una determinada marca.

2- Mostrar marca y modelo de los autos fabricados antes del 2000.

3- Generar una regla que muestre el modelo y precio de todos los autos toyota cuyo kilometraje no exceda los 100000km.

</>

Resolver TP8

} /> [

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1