



# ESTRUCTURAS DE LENGUAJES

## PROGRAMACIÓN LÓGICA

MSc. Jimena Adriana Timaná Peña

# Introducción al cálculo de predicados

# Introducción al cálculo de predicados

- El enfoque de la **Programación Lógica** consiste en expresar los programas en una forma de **lógica simbólica** y que haciendo uso de un proceso de *inferencia lógica*, logra generar resultados.

**lógica simbólica** : también conocida como lógica matemática, en la cual se estudia la inferencia mediante la construcción de sistemas formales como la lógica proposicional (las proposiciones son enunciados que tienen un valor de verdad).

La **lógica simbólica** puede ser usada para:

- Expresar proposiciones
- Expresar las relaciones entre las proposiciones
- Describir cómo nuevas proposiciones pueden ser inferidas a partir de otras proposiciones que se asumen como verdaderas.


# Introducción al cálculo de predicados

- Los lenguajes basados en **lógica simbólica** son llamados lenguajes de programación lógica o lenguajes declarativos.
- Tanto la **sintaxis** como la **semántica** de los lenguajes de programación lógica es notablemente diferente de los lenguajes imperativos, orientados a objetos y funcionales.



# Introducción al cálculo de predicados

- Tenga en cuenta que el tipo de lógica que se utiliza en la Programación Lógica es el cálculo de predicados de 1er Orden (forma de expresar de manera formal enunciados lógicos o proposiciones).



Sentencia lógica que puede o no ser verdad

- Ejemplos enunciados lógicos:
  - Un caballo es un mamífero
  - Un ser humano es un mamífero

# Introducción al cálculo de predicados

**Los siguientes enunciados son ejemplos de enunciado lógicos:**

0 es un número natural

1 es un número natural

2 es un número natural

Para todo X, si X es un número natural, entonces también lo es el sucesor de X

**Una traducción al cálculo de predicados sería:**

$\text{natural}(0)$

$\text{natural}(1)$

$\text{natural}(2)$

Para todo X,  $\text{natural}(X) \rightarrow \text{natural}(\text{sucesor}(X))$

# Introducción al cálculo de predicados

El cálculo de predicados de 1er orden clasifica las distintas partes de los enunciados de la siguiente manera:

1. Constantes
2. Variables
3. Predicados
4. Funciones
5. Conectores
6. Cuantificadores
7. Símbolos de puntuación

# Introducción al cálculo de predicados

## Clasificación partes del enunciado

### 1. Constantes

- números o nombres
- Se les llama **átomos**
- Suelen escribirse en minúscula

natural(0)  
natural(1)  
natural(2)

### 2. Variables

- Suelen escribirse en mayúscula
- Representan aquellas cantidades todavía no especificadas

Para todo  $X$ ,  $\text{natural}(X) \rightarrow \text{natural}(\text{sucesor}(X))$

### 3. Predicados

- Se utilizan para expresar propiedades de los objetos y la relación entre ellos.
- Son nombres de funciones que son V/F
- Pueden tomar varios argumentos
- En el ejemplo, el predicado *natural* toma solo un argumento



# Introducción al cálculo de predicados

## 4. Funciones

Para todo  $X$ ,  $\text{natural}(X) \rightarrow \text{natural}(\text{sucesor}(X))$

- El Cálculo de 1er orden distingue entre:
  - Funciones que son V o F, éstos son los predicados.
  - Y todas las demás funciones, representan valores no booleanos

## 5. Conectores

- Incluyen las operaciones: Y, O, NO
- Implicación “ $\rightarrow$ ”
  - $a \rightarrow b$  significa que  $b$  es verdadero, siempre que  $a$  lo sea
- Equivalencia “ $\leftrightarrow$ ”
- Orden de Precedencia: NO, Y, O,  $\leftrightarrow$ ,  $\rightarrow$

# Introducción al cálculo de predicados

## 5. Conectores

Name	Symbol	Example	Meaning
negation	$\neg$	$\neg a$	not a
conjunction	$\cap$	$a \cap b$	a and b
disjunction	$\cup$	$a \cup b$	a or b
equivalence	$\equiv$	$a \equiv b$	a is equivalent to b
implication	$\supset$	$a \supset b$	a implies b
	$\subset$	$a \subset b$	b implies a

# Introducción al cálculo de predicados

## 6. Cuantificadores

Símbolo que indica cuántos elementos que integran un conjunto dado cumplen con determinada propiedad.

- Para todo: cuantificador Universal
- Existe: cuantificador existencial

Para todo  $X$ ,  $\text{natural}(X) \rightarrow \text{natural}(\text{sucesor}(X))$

**Observación:** Una variable introducida por un cuantificador se dice que está *ligada* por el cuantificador

Name	Example	Meaning
universal	$\forall X.P$	For all $X$ , $P$ is true
existential	$\exists X.P$	There exists a value of $X$ such that $P$ is true

# Introducción al cálculo de predicados

## 7. Símbolos de puntuación

- Estos incluyen paréntesis izquierdo y derecho, la coma y el punto (este último estrictamente no es necesario).

- Ejemplos:

$\forall X.(\text{mujer}(X) \supset \text{humano}(X))$

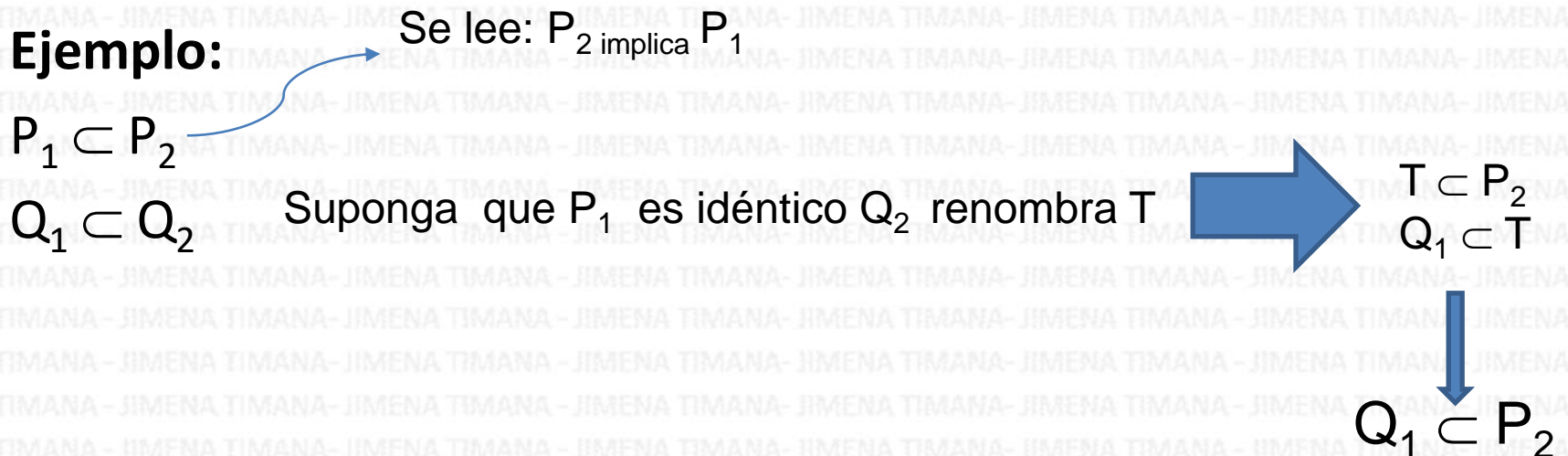
$\exists X.(\text{madre}(\text{maría}, X) \cap \text{masculino}(X))$



# Introducción al cálculo de predicados

- Concepto de **Resolución**

Es una regla de inferencia que permite inferir proposiciones que se han calculado a partir de proposiciones dadas.



# Introducción al cálculo de predicados

- Para simplificar la manera de expresar los predicados o proposiciones , se puede usar la forma de **cláusula**.

**Ejercicio:** Considere las siguientes dos proposiciones expresadas en cláusulas:

$\text{mucho\_mayor}(\text{maria}, \text{jimena}) \subset \text{madre}(\text{maria}, \text{jimena})$

$\text{mas\_sabio}(\text{maria}, \text{jimena}) \subset \text{mucho\_mayor}(\text{maria}, \text{jimena})$

1. Realice la lectura de las proposiciones.
2. Construya una proposición usando resolución.

# Introducción al cálculo de predicados

**Solución:**

mucho\_mayor(maria, jimena)  $\subset$  madre(maria, jimena)

mas\_sabio(maria, jimena)  $\subset$  mucho\_mayor(maria, jimena)

$\top$   $\subset$  madre(maria, jimena)

mas\_sabio(maria, jimena)  $\subset$   $\top$

mas\_sabio(maria, jimena)  $\subset$  madre(maria, jimena)

# Introducción al cálculo de predicados

Otra forma es combinando los lados izquierdos y derechos de ambas proposiciones y después cancelar aquellos enunciados que coinciden en ambos lados.

$\text{mucho\_mayor}(\text{maria}, \text{jimena}) \subset \text{madre}(\text{maria}, \text{jimena})$

$\text{mas\_sabio}(\text{maria}, \text{jimena}) \subset \text{mucho\_mayor}(\text{maria}, \text{jimena})$

$\text{mucho\_mayor}(\text{maria}, \text{jimena}), \text{mas\_sabio}(\text{maria}, \text{jimena}) \subset \text{madre}(\text{maria}, \text{jimena}), \text{mucho\_mayor}(\text{maria}, \text{jimena})$

~~$\text{mucho\_mayor}(\text{maria}, \text{jimena}), \text{mas\_sabio}(\text{maria}, \text{jimena}) \subset \text{madre}(\text{maria}, \text{jimena}), \text{mucho\_mayor}(\text{maria}, \text{jimena})$~~

$\text{mas\_sabio}(\text{maria}, \text{jimena}) \subset \text{madre}(\text{maria}, \text{jimena})$



# Introducción al cálculo de predicados

La **Resolución** en realidad puede ser mucho más compleja.

El uso de variables en las proposiciones obliga a encontrar valores para esas variables de forma que se produce una verdad.

# Introducción al cálculo de predicados

## Concepto de Unificación:

Encontrar los valores para las variables en las proposiciones de modo que los enunciados se hagan idénticos y se de así un proceso de pareamiento o *matching*

## Concepto de Instanciación:

Asignación temporal de valores a las variables para permitir que el proceso de Unificación tenga éxito.

# Introducción al cálculo de predicados

## Ejemplo:

humano(carlos).

humano(matias).

planeta(matias, tierra).

$\text{mortal}(X) \subset \text{humano}(X) \cap \text{planeta}(X, \text{tierra}).$

Lectura:

*Si X es humano y si ese X vive en el planeta tierra implica que X es mortal.*

## Unificación

Encontrar los **valores** para las variables en las proposiciones de modo que los enunciados se hagan idénticos y se de así un proceso de pareamiento o *matching*

## Instanciación

Asignación temporal de valores a las variables para permitir que el proceso de Unificación tenga éxito.

Después de instanciar una variable con un valor, si el pareamiento o matching falla, puede ser necesario dar marcha atrás e instanciar con un valor diferente. A este proceso se le denomina **Backtracking**

# **Visión general de la Programación Lógica**



# Visión general de la Programación Lógica

- Los lenguajes que usan la programación lógica son llamados **Lenguajes Declarativos** porque dichos programas se componen de declaraciones (proposiciones, en lógica simbólica) en lugar de asignaciones y sentencias de control de flujo.

# Visión general de la Programación Lógica

Una de las características esenciales de la Programación Lógica es su semántica, la cual es llamada **semántica declarativa**.

- **Semántica declarativa**

- el significado de una proposición dada en un lenguaje de programación lógica puede ser directamente determinada a partir de la sentencia misma.
- Mucho más simple que la semántica de los lenguajes imperativos (requiere examinación de variables, tipos de datos, etc).

# Elementos básicos de Prolog

# El lenguaje Prolog

## Historia

- El lenguaje de programación PROLOG (“PROgrammation en LOGique” o Programación Lógica) fue creado por Alain Colmerauer y sus colaboradores alrededor de 1970 en la Universidad de Marseille-Aix para hacer deducciones a partir de texto.
- A diferencia de otros lenguajes, Prolog no es un lenguaje de programación para usos generales, sino de propósito específico, que está orientado a resolver problemas usando el cálculo de predicados.
- Es considerado el lenguaje principal de la Programación Lógica.
- El lenguaje PROLOG juega un importante papel dentro de la Inteligencia Artificial → Sistemas Expertos



# El lenguaje Prolog

## Historia

PROLOG es un LP se utiliza para resolver problemas en los que entran en juego: *objetos y relaciones* entre ellos.

### Ejemplo:

“matías tiene una bicicleta”

- Se expresa una relación entre un objeto (matías) y otro objeto en particular (bicicleta).
- Relaciones con orden específico (matías tiene una bicicleta y no al contrario).

Pregunta: ¿Tiene matías una bicicleta? lo que estamos haciendo es indagando acerca de una relación.

# Objetos de datos

## Tipos de datos primitivos soportados

- enteros(1,2,3, ...)
- reales(1.2, 3.4, 5.9, ...)
- caracteres('a', 'b'. ....)

Nota: para los caracteres y cadenas no es necesario las comillas.

Las variables deben escribirse en mayúsculas y las constantes en minúsculas.

# Objetos de datos

## Tipos de datos estructurados

- Los objetos pueden ser átomos o listas
- Listas se representan como [A,B,C,...]
- La notación [A|B] se usa para indicar que A es la cabeza de la lista y B la cola de la lista.

## Tipos de datos definidos por el usuario

- No existen verdaderos tipos de datos definidos por el usuario.
- Las reglas “pueden” actuar como si lo fueran.

# Estructura de un programa en Prolog

En esencia, un programa en PROLOG se compone de una serie de **hechos, reglas y preguntas**.

Los hechos y las reglas se usan para determinar cuáles sustituciones de variables de la pregunta son congruentes con los datos almacenados.



# Hechos

- Se utilizan para expresar propiedades de los objetos (átomos o constantes) y las relaciones entre ellos.
- Son tuplas  $n$   $f(a_1, a_2, \dots, a_n)$  y expresan la relación entre los  $n$  argumentos de acuerdo con la relación  $f$ .

## Ejemplos:

mujer(jimena).

mama(jimena, matias).

mama(maria, jimena).

abuela(maria, matias).

estudiante(jimena, timaná, femenino, 4.2).

# Reglas

- Las reglas se ven como implicaciones o condicionales.
- En Prolog se usa el símbolo :- para representar lo que llamamos una **regla**:

*cabeza\_de\_la\_regla :- cuerpo\_de\_la\_regla.*

El símbolo :- significa “sí”, o si lo leemos de derecha a izquierda, “entonces” o “implica”

Ejemplo:

mortal(X):-humano(X).

- Se lee X es mortal sí X es humano. El símbolo :- significa “sí”, o si lo leemos de derecha a izquierda, “entonces” o “implica”. En esta regla, mortal(X) es la cabeza, y humano(X) es el cuerpo.

# Preguntas

- Las preguntas son las herramientas que tenemos para recuperar la información desde Prolog. Al hacer una pregunta a un programa lógico queremos determinar si esa pregunta es *consecuencia lógica* del programa.
- Cuando a Prolog se le hace una pregunta con una variable, dicha variable estará inicialmente no instanciada.
- Prolog entonces busca de un hecho que *empareje* con la pregunta: los símbolos de predicado y el número de argumentos sean iguales, y emparejen los argumentos. Entonces Prolog hará que la variable se instancie con el argumento que esté en su misma posición en el hecho.
- Cuando encuentra un hecho que empareje, saca por pantalla los objetos que representan ahora las variables

# Funciones aritméticas

- Si queremos hacer una asignación:

?- X is 3+5, write(X), nl.      Donde *nl* es un salto de línea.

## Algunas funciones aritméticas incorporadas en Prolog:

- abs, sig, min, max, random, round, sqrt, sin, cos, tan, log, log10, exp, entre otras.

?- X is sqrt(144), write(X), nl.

- **Nota:** si la respuesta genera varios resultados y se quiere listarlos o visualizarlo se deberá usar *ln*.



# Operadores Matemáticos

**+** Suma

**-** Resta

**\*** Multiplicación

**/** División (retorna siempre en punto flotante)

**//** División entera (trunca)

**mod** Resto de división

**\*\*** Potenciación

# Operadores Relacionales

> Mayor que

< Menor que

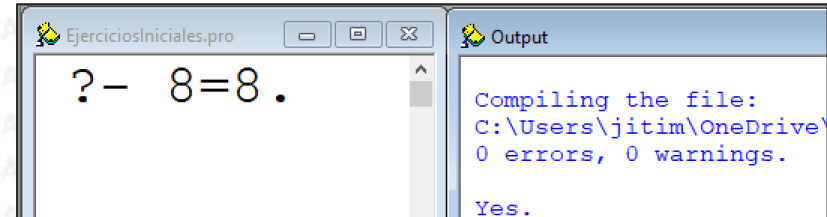
>= Mayor o igual que

=< Menor o igual que

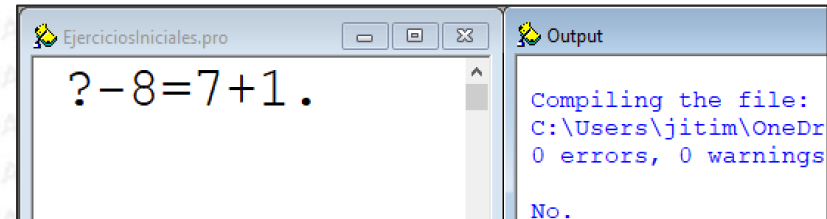
== Aritméticamente igual

!= Aritméticamente diferente

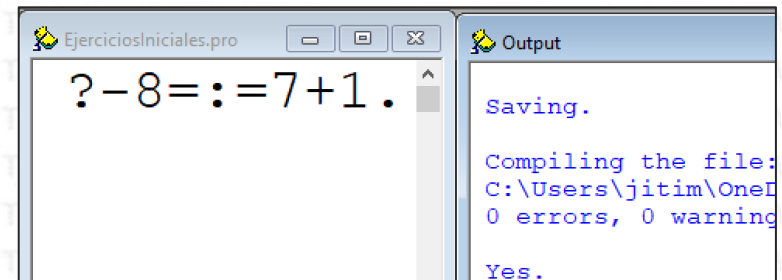
- Nota: el *igual* = compara dos términos, y es verdadero solamente cuando ambos son **idénticos**.(palabras y números)



The screenshot shows a code editor window titled 'EjerciciosIniciales.pro' with the text '? - 8 = 8 .' entered. To the right is an 'Output' window showing the compilation process: 'Compiling the file: C:\Users\jitim\OneDrive\0 errors, 0 warnings. Yes.'



The screenshot shows a code editor window titled 'EjerciciosIniciales.pro' with the text '? - 8 = 7 + 1 .' entered. To the right is an 'Output' window showing the compilation process: 'Compiling the file: C:\Users\jitim\OneDrive\0 errors, 0 warnings No.'



The screenshot shows a code editor window titled 'EjerciciosIniciales.pro' with the text '? - 8 == 7 + 1 .' entered. To the right is an 'Output' window showing the compilation process: 'Saving. Compiling the file: C:\Users\jitim\OneDrive\0 errors, 0 warnings Yes.'

# Ejemplo

## Ejemplo 1:

- Cree en Prolog unos **hechos** que le permitan indicar que felipe, matias, joaquin y luisa son humanos.
- Ahora cree una **regla** que permita representar:
  - X es mortal sí X es humano ( si se lee de izquierda a derecha) o
  - X es humano entonces X es mortal ( si se lee de derecha a izquierda)
- Realice las siguientes **preguntas**:
  - ¿felipe es humano?
  - ¿Quién o quiénes son humanos?
  - ¿Quién o quiénes son mortales?
  - ¿Es carlos mortal?
  - ¿Es joaquin mortal?

# Ejercicios preliminares

- Deduzca el estado del tiempo a partir de cómo se encuentra el suelo.
- Determine si un número es par.
- Un alumno gana la asignatura si la nota definitiva es mayor o igual a 3.