TEMA 1: LÓGICA Y PROGRAMACIÓN

1. Introducción
2. Lógica proposicional
3. Lógica de predicados
4. ➡ **4. Programación lógica**
    - Algoritmo de normalización clausal
    - Unificación
    - Resolución

4. Programación lógica

- La programación lógica nace de la idea de crear un lenguaje más cercano al lenguaje natural y a la forma de pensar humana.
- Se intenta separar la lógica del problema del control del programa:

  Algoritmo = Lógica + Control
- Sin embargo, Gödel demostró que la lógica no es decidible, por tanto no se puede utilizar como lenguaje de programación.

4. Programación lógica

- En programación lógica se utiliza un **subconjunto** de la lógica de predicados que sí que es decidible y además se resuelve de manera eficiente.
- Únicamente se utilizan los operadores $\lor$ y $\sim$ y el cuantificador universal $\forall$.
- Como regla de inferencia se utiliza únicamente la **regla de resolución**, que es una generalización de la regla de EI.
- La regla de resolución necesita además la **unificación**, que consiste en unificar en un sólo literal, dos literales con variables.

4.1. Algoritmo de normalización clausal

- Transforma fórmulas de lógica de predicados en **fórmulas clausales** o **cláusulas**.
- Como ejemplo utilizaremos la siguiente frase:

  "La sucesión de números primos está acotada y no está acotada"

En lógica de predicados:

p(X): X es primo

m(Y,X): Y>X

$$\exists X: (p(X) \land \forall Y: (p(Y) \Rightarrow \sim m(Y,X))) \land$$
$$\sim\exists X: (p(X) \land \forall Y: (p(Y) \Rightarrow \sim m(Y,X)))$$

4.1. Algoritmo de normalización clausal

1. **Eliminar los símbolo de implicación**
    - Se eliminan utilizando las equivalencias:
        - $p \Leftrightarrow q \equiv (p \Rightarrow q) \land (q \Rightarrow p)$
        - $p \Rightarrow q \equiv \sim p \lor q$

$$\exists X: (p(X) \land \forall Y: (p(Y) \Rightarrow \sim m(Y,X))) \land$$
$$\sim\exists X: (p(X) \land \forall Y: (p(Y) \Rightarrow \sim m(Y,X)))$$
$$\Downarrow$$
$$\exists X: (p(X) \land \forall Y: (\sim p(Y) \lor \sim m(Y,X))) \land$$
$$\sim\exists X: (p(X) \land \forall Y: (\sim p(Y) \lor \sim m(Y,X)))$$

4.1. Algoritmo de normalización clausal

2. **Reducir negaciones a fórmulas atómicas**
    - Se realiza utilizando las leyes de De Morgan y las leyes de De Morgan generalizadas:
        - $\sim (p \land q) \equiv \sim p \lor \sim q$ $\quad$ $\sim (p \lor q) \equiv \sim p \land \sim q$
        - $\sim\forall\ i: P(i) \equiv \exists\ i: \sim P(i)$ $\quad$ $\sim \exists\ i: P(i) \equiv \forall\ i: \sim P(i)$

$$\exists X: (p(X) \land \forall Y: (\sim p(Y) \lor \sim m(Y,X))) \land$$
$$\sim\exists X: (p(X) \land \forall Y: (\sim p(Y) \lor \sim m(Y,X)))$$
$$\Downarrow$$
$$\exists X: (p(X) \land \forall Y: (\sim p(Y) \lor \sim m(Y,X))) \land$$
$$\forall X: (\sim p(X) \lor \exists Y: (p(Y) \land m(Y,X)))$$

4.1. Algoritmo de normalización clausal

3. **Renombrar las variables duplicadas**
   - Renombramos variables para evitar confusiones cuando eliminemos cuantificadores:

∃X: (p(X) ∧ ∀Y: (~p(Y) ∨ ~m(Y,X))) ∧
∀X: (~p(X) ∨ ∃Y: (p(Y) ∧ m(Y,X)))

⬇

∃U: (p(U) ∧ ∀V: (~p(V) ∨ ~m(V, U))) ∧
∀X: (~p(X) ∨ ∃Y: (p(Y) ∧ m(Y,X)))

4.1. Algoritmo de normalización clausal

4. **Skolemizar la fórmula**
   - Eliminar los cuantificadores existenciales mediante funciones de Skolem.
   - **Constantes de Skolem:** Si el cuantificador existencial es el primer cuantificador que aparece en la fórmula (no depende de otro cuantificador), podemos eliminarlo sustituyendo la variable por una constante **nueva** (regla de EP).

Ejemplo:

Existe una mujer cuya madre es Eva:
∃ X (mujer(X) ∧ madre(eva, X))

Existe un país donde todos sus habitantes son felices
∃ Y: ∀ X: (habitante(X,Y) ⇒ feliz(X))

4.1. Algoritmo de normalización clausal

- **Funciones de Skolem:** Si el cuantificador existencial depende de uno o varios cuantificadores universales, podemos eliminarlo sustituyendo la variable por una función **nueva** que dependa de las variables de los cuantificadores universales.
- La constante de Skolem es una función de Skolem sin parámetros.

Ejemplo:

Para todo entero X, existe un entero Y mayor que X:
∀ X: ∃ Y: mayor(Y, X)

∀ X, Y: (∃ Z: (∀ U, V: (∃ W: p(W,Z))))

4.1. Algoritmo de normalización clausal

4. **Skolemizar la fórmula**

∃U: (p(U) ∧ ∀V: (~p(V) ∨ ~m(V, U))) ∧
∀X: (~p(X) ∨ ∃Y: (p(Y) ∧ m(Y,X)))

⬇

p(h) ∧ ∀V: (~p(V) ∨ ~m(V, h)) ∧
∀X: (~p(X) ∨ (p(f(X)) ∧ m(f(X), X)))

4.1. Algoritmo de normalización clausal

5. **Desplazar los cuantificadores universales a la izquierda**
   - Se desplazan todos los cuantificadores a la izquierda. La expresión resultante se dice que está en **forma prenex**

p(h) ∧ ∀V: (~p(V) ∨ ~m(V, h)) ∧
∀X: (~p(X) ∨ (p(f(X)) ∧ m(f(X), X)))

⬇

∀V: ∀X: p(h) ∧ (~p(V) ∨ ~m(V, h)) ∧
(~p(X) ∨ (p(f(X)) ∧ m(f(X), X)))

4.1. Algoritmo de normalización clausal

6. **Eliminar los cuantificadores universales**
   - Toda variable existente en la fórmula tiene un cuantificador universal a la izquierda, por tanto los cuantificadores son redundantes y se eliminan por convención de notación.

∀V: ∀X: p(h) ∧ (~p(V) ∨ ~m(V, h)) ∧
(~p(X) ∨ (p(f(X)) ∧ m(f(X), X)))

⬇

p(h) ∧ (~p(V) ∨ ~m(V, h)) ∧ (~p(X) ∨ (p(f(X)) ∧ m(f(X), X)))

4.1. Algoritmo de normalización clausal

7. **Convertir la fórmula a su FNC**
   - Convertir la fórmula a la *Forma Normal Conjuntiva* (FNC), que consiste en conjunción de disyunciones.
   - Utilizamos la equivalencia:

$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$$

$$p(h) \wedge (\sim p(V) \vee \sim m(V, h)) \wedge (\sim p(X) \vee (p(f(X)) \wedge m(f(X), X)))$$

⬇

$$p(h) \wedge (\sim p(V) \vee \sim m(V, h)) \wedge (\sim p(X) \vee p(f(X))) \wedge (\sim p(X) \vee m(f(X), X))$$

4.1. Algoritmo de normalización clausal

8. **Eliminar las conjunciones. Forma clausal.**
   - Afirmar la conjunción de varias fórmulas es como afirmar las fórmulas por separado, por lo que se eliminan las conjunciones separando las fórmulas.
   - Las fórmulas resultantes se denominan *cláusulas*:

$$p(h) \wedge (\sim p(V) \vee \sim m(V, h)) \wedge (\sim p(X) \vee p(f(X))) \wedge (\sim p(X) \vee m(f(X), X))$$

$$\left.\begin{array}{l} p(h) \\ \sim p(V) \vee \sim m(V, h) \\ \sim p(X) \vee p(f(X)) \\ \sim p(X) \vee m(f(X), X) \end{array}\right\} \textbf{Cláusulas}$$

4.1. Algoritmo de normalización clausal

Reescritura utilizando la implicación.

- Las cláusulas se pueden escribir utilizando la implicación (⇒) y la conjunción (∧), ya que queda más claro su significado y es como las utilizaremos en los programas.
- Los literales negativos pasan a ser antecedentes y los positivos consecuentes: $p \Rightarrow q \equiv \sim p \vee q$

p(h) ⇐	p(h).	**Programa PROLOG**
⇐ p(V), m(V, h)	:- p(V), m(V, h)	
p(f(X)) ⇐ p(X)	p(f(X)) :- p(X)	
m(f(X), X) ⇐ p(X)	m(f(X), X) :- p(X)	

4.1. Algoritmo de normalización clausal

- **Cláusulas de Horn.**
  - Las cláusulas se pueden clasificar atendiendo al tipo de literales que contienen.

Cláusulas
- **0 lit. +**
  - **0 lit. -** ⇐ ≡ F
  - **1 o + lit. -** **Pregunta**
- **1 lit. +**
  - **0 lit. -** **Hecho**
  - **1 o + lit. -** **Regla**
- **2 o + lit. +** **Cláusulas no Horn**

Cláusulas de Horn

4.1. Algoritmo de normalización clausal

- Clasificación de las cláusulas del ejemplo anterior:

p(h) ⇐	Hecho
⇐ p(V), m(V, h)	Pregunta
p(f(X)) ⇐ p(X)	Regla
m(f(X), X) ⇐ p(X)	Regla

- **Ejemplo de cláusula no Horn:**

  (m(X) ∨ q(X)) ⇐ p(X)