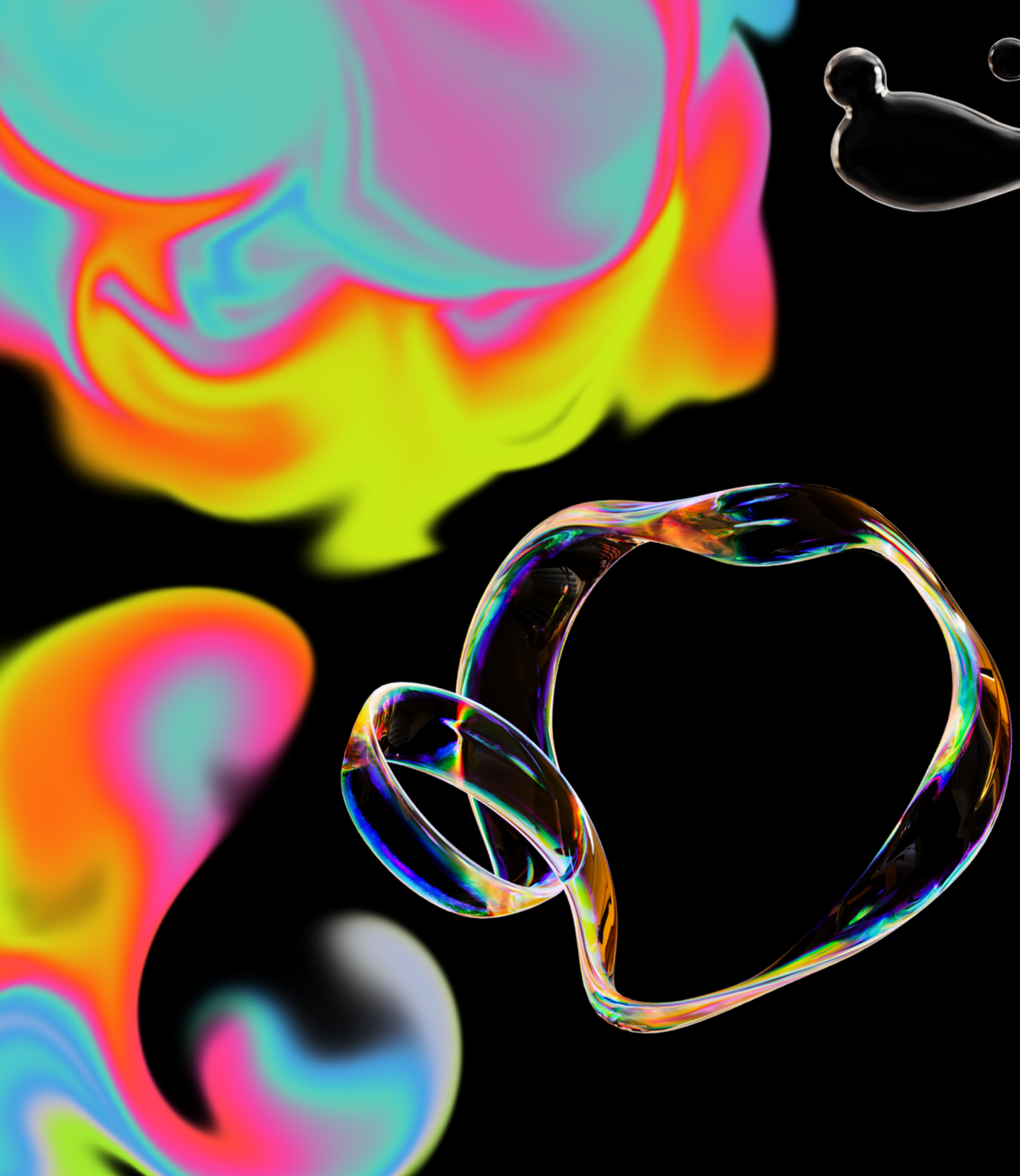


Paradigma: Programación Lógica

MARIA FERNANDA CABRALES
JUAN PABLO CASTRO AVILA
JUAN DAVID SANTAMARIA
TANIA GISSELL QUIJANO

LENGUAJES DE PROGRAMACIÓN

PARADIGMAS DE PROGRAMACIÓN

Clasificaciones de los lenguajes de programación según sus características. Existen dos grandes grupos:

1. Programación imperativa
   ↓
   ¿Cómo hacer las cosas?

2. Programación declarativa
   ↓
   ¿Qué cosas hacer?

PARADIGMAS DE PROGRAMACIÓN

1. Programación imperativa

↓

¿Cómo hacer las cosas?

↓

```
> Sacar dos rodajas de pan
> Sacar jamón y queso
> Poner una rodaja de pan
> Poner una rodaja de jamón sobre esa
  rodaja de pan
 ...
```

2. Programación declarativa

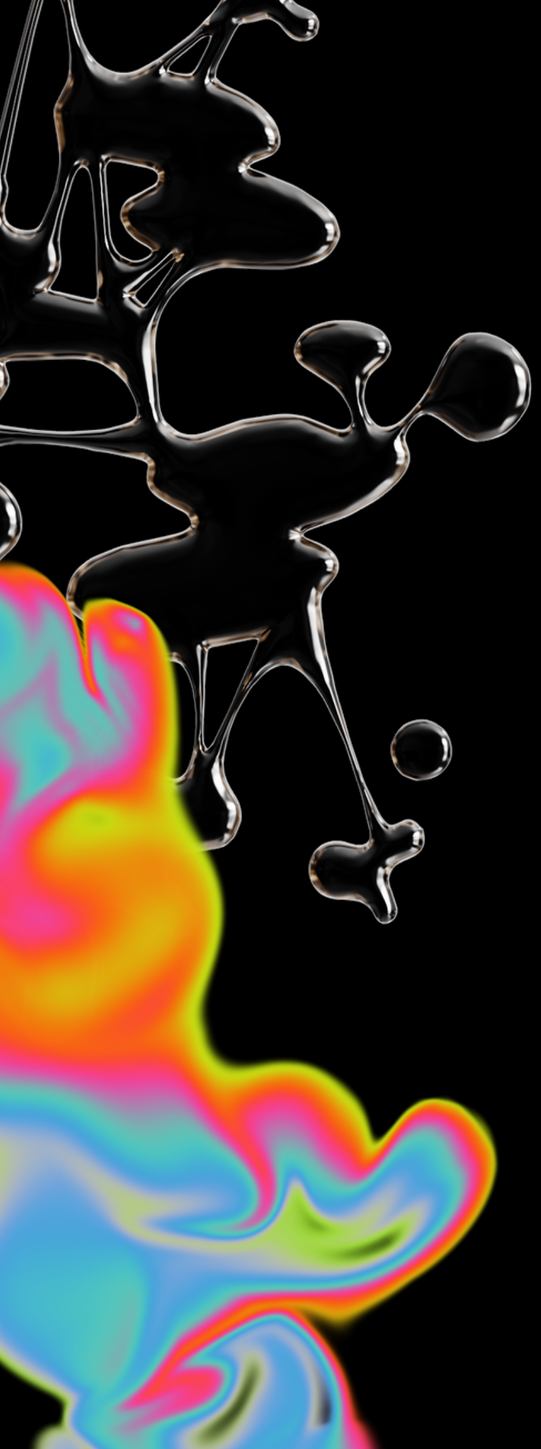
↓

¿Qué cosas hacer?

↓

```
> Un emparedado tiene un pan, encima va
  una lechuga, luego una rodaja de
  tomate y otra de queso, finalmente
  otra rodaja de pan.

> Haz un emparedado
```



PARADIGMAS DE PROGRAMACIÓN

2. Programación declarativa

↓

¿Qué cosas hacer?

↓

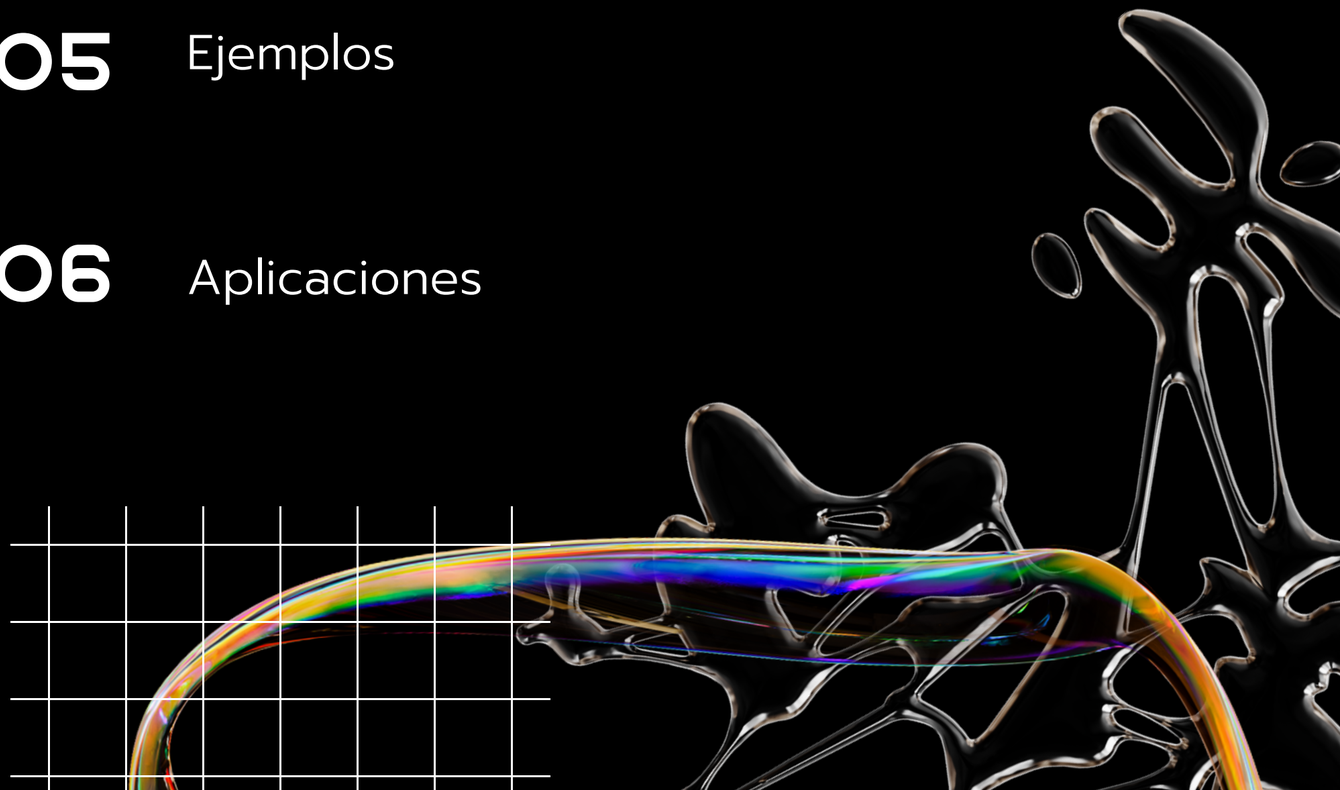
```
> Un emparedado tiene un pan, encima va
  una lechuga, luego una rodaja de
  tomate y otra de queso, finalmente
  otra rodaja de pan.

> Haz un emparedado
```

↓

Programación Lógica

Contenido



FILOSOFIA DEL PARADIGMA

1) ¿Qué busca?
Modelar problemas abstractos.

2) ¿Cómo lo hace?
Utilizando un conjunto de reglas,
predicados, hechos lógicos y sus relaciones.

3) ¿Qué logra?
Llegar a conclusiones por medio de la
inducción.

¿Por qué?
Resolución de enunciados lógico-matemáticos
y demostración de teoremas

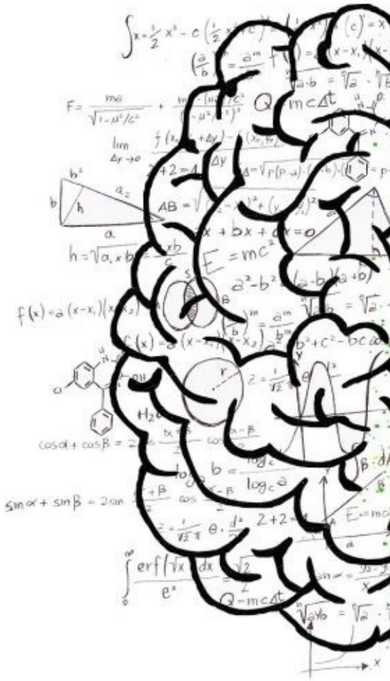
FILOSOFIA DEL PARADIGMA

algoritmos = lógica + control

Ambigüedades

Axiomas

Inferencia



>Hecho #1: Juanito es hijo de Juan

>Hecho #2: Marianita es hija de Mariana

>Hecho #3: Juanita es hermana de Juanito

Pregunta: ¿Existe un X que sea hijo de Juan?

Respuesta:

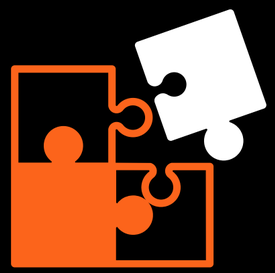
>Juanito
>Juanita

CONCEPTOS CLAVE

Conocer y entender la terminología usada anteriormente es vital para el completo entendimiento de cómo funciona este paradigma

VENTAJAS

1. Expresión Clara del Conocimiento
2. Desarrollo Rápido
3. Separación entre Datos y Programa
4. Razonamiento Automático
5. Resolución de Problemas Complejos
6. Abstracción Alta
7. Independencia de la Arquitectura de la Máquina
8. Prototipado
9. Representación y Consulta Eficaz de Datos
10. Facilitación de la Inteligencia Artificial



DESVENTAJAS

1. Eficiencia
2. Áreas de Aplicación Limitadas
3. Herramientas de Depuración Escasas
4. Curva de Aprendizaje
5. Poca Adopción en la Industria
6. Respuestas Ambiguas
7. Dificultad para Representar Algunos Conceptos
8. Inferencia Limitada por la Base de Conocimiento
9. Desarrollo de Interfaz de Usuario
10. Desempeño en Tiempo Real

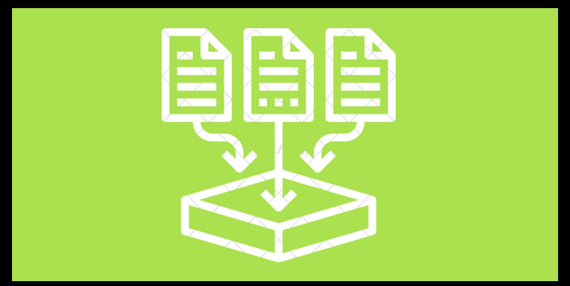
LENGUAJES DE PROGRAMACIÓN

Prolog

Datalog

Mercury

Absys

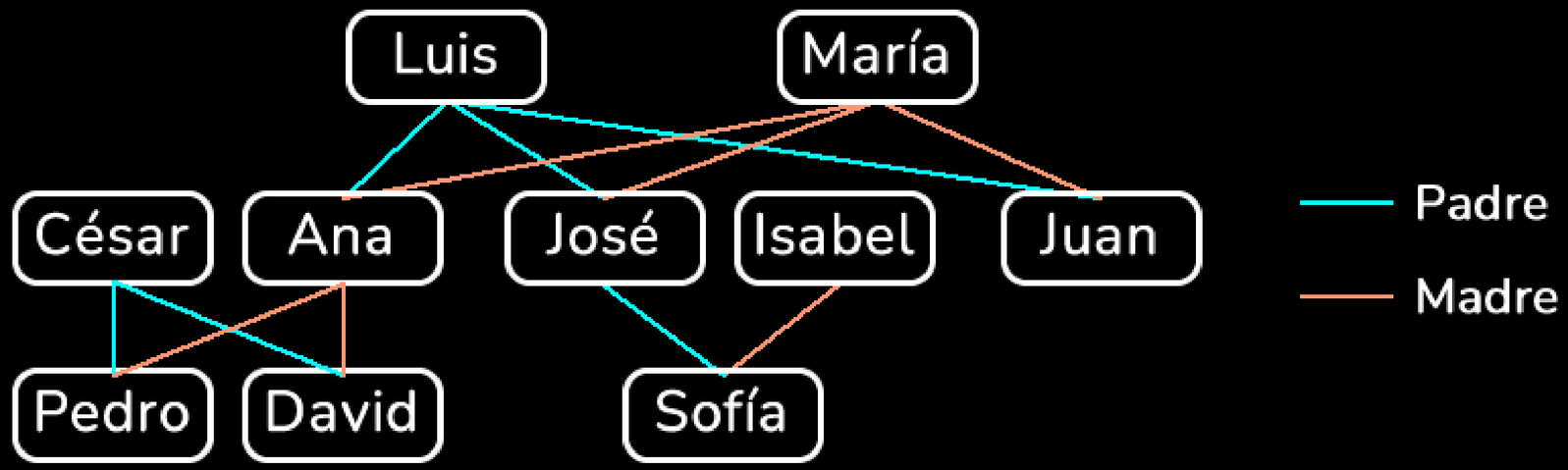


Vadalog

CycL

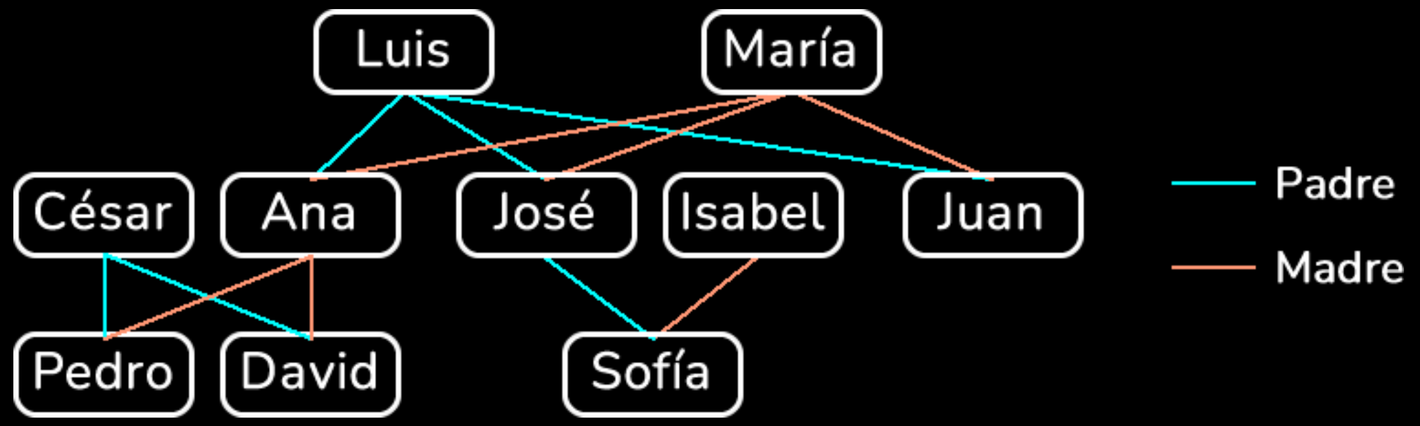
EJEMPLOS EN DISTINTOS LENGUAJES

Como primer ejemplo, veremos la implementación de un árbol familiar o genealógico por medio de diferentes lenguajes de programación lógica.



PROLOG

```
% Hechos
padre(luis, ana).
padre(luis, jose).
padre(luis, juan).
padre(cesar, pedro).
padre(cesar, david).
padre(jose, sofia).
madre(maria, ana).
madre(maria, jose).
madre(maria, juan).
madre(ana, pedro).
madre(ana, david).
madre(isabel, sofia).
```



PROLOG

```
% Reglas: Definición de relaciones familiares

hermano(X, Y) :-
    padre(Z, X),
    padre(Z, Y),
    madre(W, X),
    madre(W, Y),
    X \= Y.

abuelo(Abuelo, Nieto) :-
    padre(Abuelo, Hijo),
    padre(Hijo, Nieto);
    padre(Abuelo, Hijo),
    madre(Hijo, Nieto).

abuela(Abuela, Nieto) :-
    madre(Abuela, Hijo),
    madre(Hijo, Nieto);
    madre(Abuela, Hijo),
    padre(Hijo, Nieto).
```

PROLOG

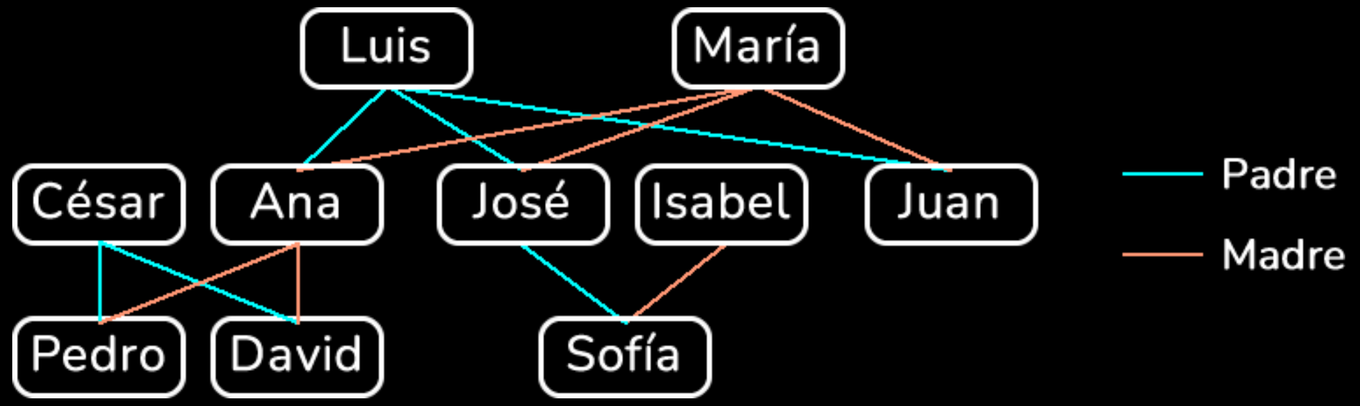
```
%Consultas

hermano(david, pedro)
> true

hermano(sofia, pedro)
> false
```

```
% Reglas:

hermano(X, Y) :-
    padre(Z, X),
    padre(Z, Y),
    madre(W, X),
    madre(W, Y),
    X \= Y.
```

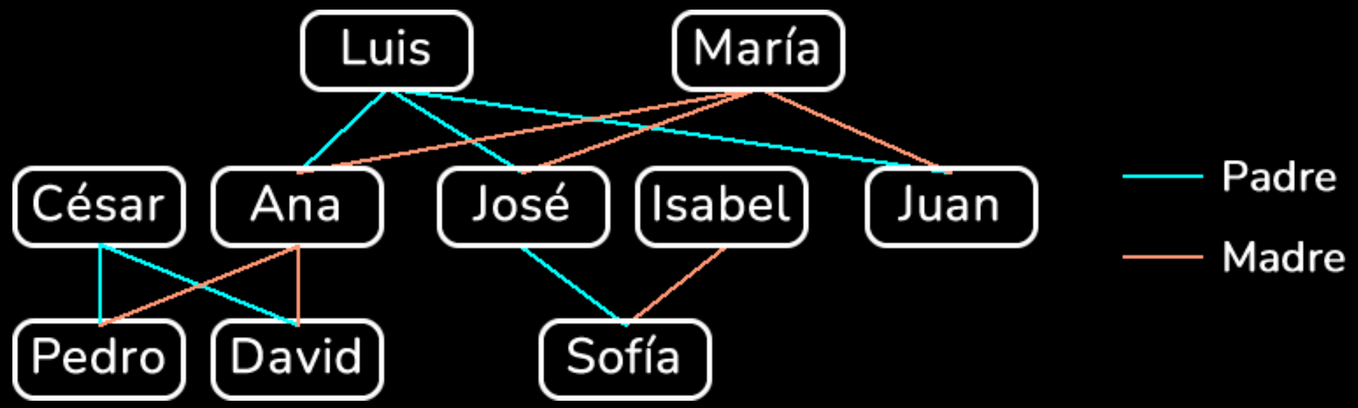


PROLOG

```
abuelo(luis, david)
> true

abuela(maria, juan)
> false

abuela(luis, pedro)
> false
```




```
abuelo(Abuelo, Nieto) :-
    padre(Abuelo, Hijo),
    padre(Hijo, Nieto);
    padre(Abuelo, Hijo),
    madre(Hijo, Nieto).

abuela(Abuela, Nieto) :-
    madre(Abuela, Hijo),
    madre(Hijo, Nieto);
    madre(Abuela, Hijo),
    padre(Hijo, Nieto).
```

MERCURY

Nombre del módulo

```
:- module relaciones_familiares.
```

```
:- interface.
```

Se importa módulo para uso de listas

```
:- import_module list.
```

```
:- type padre ---> luis; cesar; jose.
```

```
:- type madre ---> maria; ana; isabel.
```

Definición de tipos

```
:- type persona ---> luis; cesar; jose; maria; ana; isabel;
                     juan; pedro; david; sofia.
```

```
:- type relacion ---> padre(persona, persona);
                      madre(persona, persona).
```

MERCURY

Definición de funciones

```
:- func relaciones_familiares = list(relacion).
relaciones_familiares =
    [
        padre(luis, ana), padre(luis, jose), padre(luis, juan),
        padre(cesar, pedro), padre(cesar, david),
        padre(jose, sofia),
        madre(maria, ana), madre(maria, jose), madre(maria, juan),
        madre(ana, pedro), madre(ana, david),
        madre(isabel, sofia)
    ].

:- func hermano(persona, persona) = bool.
hermano(X, Y) :-
    padre(Z, X), padre(Z, Y),
    madre(W, X), madre(W, Y),
    X \= Y.
```

MERCURY

Definición de funciones

```
:- func abuelo(persona, persona) = bool.
abuelo(Abuelo, Nieto) :-
    ( padre(Abuelo, Hijo), padre(Hijo, Nieto) ;
      padre(Abuelo, Hijo), madre(Hijo, Nieto) ).

:- func abuela(persona, persona) = bool.
abuela(Abuela, Nieto) :-
    ( madre(Abuela, Hijo), madre(Hijo, Nieto) ;
      madre(Abuela, Hijo), padre(Hijo, Nieto) ).

:- implementation.
:- import_module string.
```

CYCL

```
; Definición de relaciones familiares

(#$isa #$Luis #$Padre)
(#$isa #$Cesar #$Padre)
(#$isa #$Jose #$Padre)
(#$isa #$Maria #$Madre)
(#$isa #$Ana #$Madre)
(#$isa #$Isabel #$Madre)

(#$padreDe #$Luis #$Ana)
(#$padreDe #$Luis #$Jose)
(#$padreDe #$Luis #$Juan)
(#$padreDe #$Cesar #$Santiago)
(#$padreDe #$Cesar #$David)
(#$padreDe #$Jose #$Sofia)

(#$madreDe #$Maria #$Ana)
(#$madreDe #$Maria #$Jose)
(#$madreDe #$Maria #$Juan)
(#$madreDe #$Ana #$Santiago)
(#$madreDe #$Ana #$David)
(#$madreDe #$Isabel #$Sofia)
```

CYCL

```
; Reglas de relaciones familiares

(#$and
  (#$hermanoDe ?X ?Y)
  (#$padreDe ?Z ?X)
  (#$padreDe ?Z ?Y)
  (#$madreDe ?W ?X)
  (#$madreDe ?W ?Y)
  (#$notEqual ?X ?Y)
)
```

```
(#$or
  (#$abueloDe ?Abuelo ?Nieto)
  (#$and
    (#$padreDe ?Abuelo ?Hijo)
    (#$padreDe ?Hijo ?Nieto)
  )
  (#$and
    (#$padreDe ?Abuelo ?Hijo)
    (#$madreDe ?Hijo ?Nieto)
  )
)
```

PROLOG

También es posible la solución de problemas utilizando otras técnicas, como es el caso de la recursión en Prolog. El siguiente ejemplo muestra como se calcula el factorial de un número.

```
factorial(0, 1).  % El factorial de 0 es 1.

factorial(N, Result) :-
    N > 0,
    Prev is N - 1,
    factorial(Prev, PrevFactorial),
    Result is N * PrevFactorial.
```

APLICACIONES PROGRAMACIÓN LOGICA

La Programación Lógica cuenta con un amplio rango de aplicabilidad gracias a su naturaleza basada en la unificación de la programación declarativa clásica con los conceptos de lógica formal.

Algunas aplicaciones de la Programación Lógica comúnmente utilizadas:

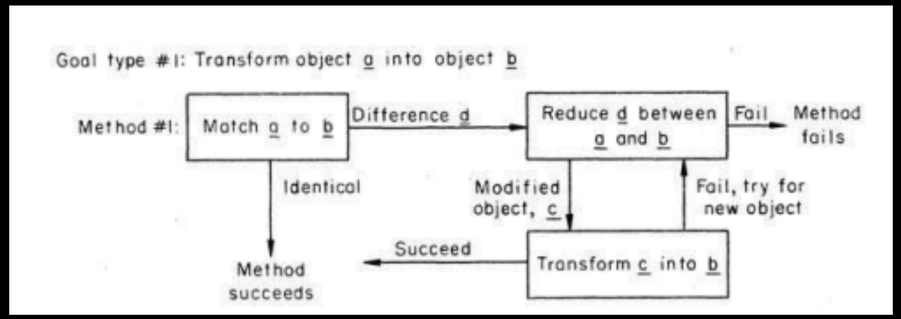
- IA y Sistemas Expertos
- Programacion Logica Probabilistica

Bases de Datos Relacionales, Negocio y comercio, Medicina, Matemáticas, Servicios Geológicos y Meteorológicos, Sociología, Lingüística Computacional, Diseño y Simulación, Gestion Documental

IA / Sistemas Expertos

- Razonamiento Lógico
- Solución de Problemas
- Reconocimiento de Lenguaje Natural

General-purpose Problem Solver (Newell, 1958)



TIPOS DE SISTEMAS EXPERTOS

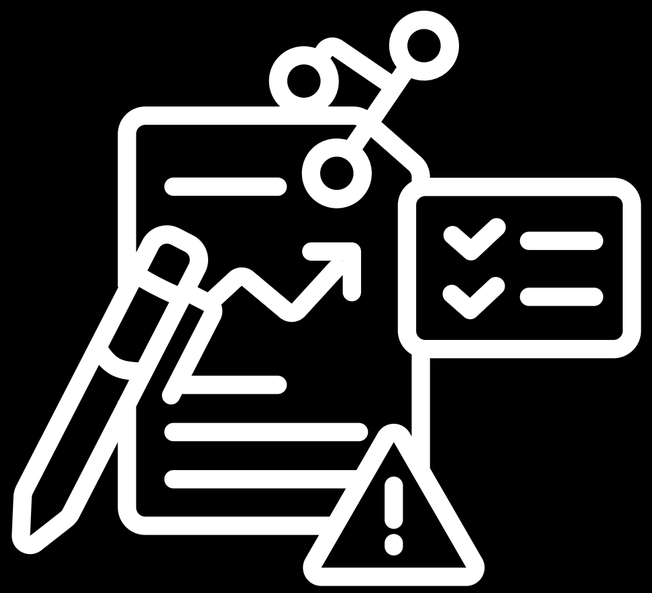
- RBR (Rule Based Reasoning)
- CBR (Case Based Reasoning)
- Redes Bayesianas

TAREAS

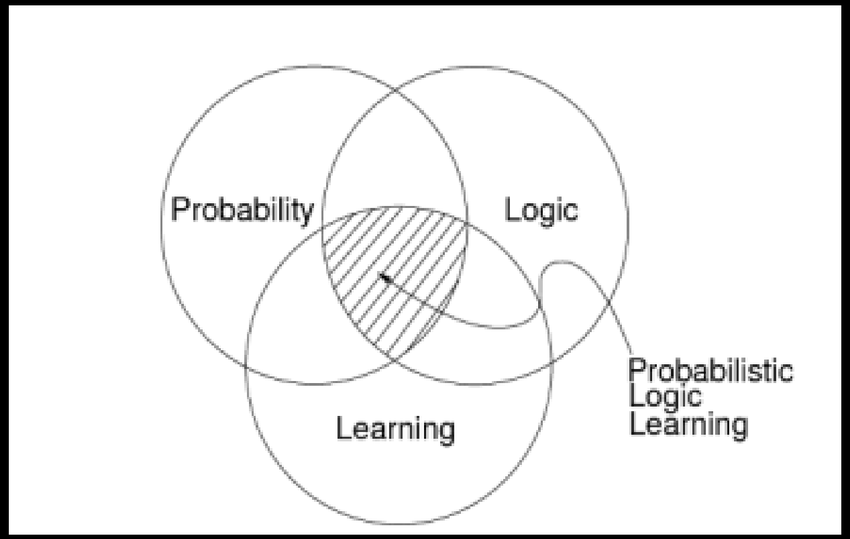
- MONITORIZACION
- DISEÑO
- CONTROL
- SIMULACION

Programación Lógica Probabilística

PROBABILIDAD



- Toma de Decisiones en Entornos de Incertidumbre
- Reconocimiento de actividades en el Tiempo
- Generación de Predicciones



Problog2 ***Vision Artificial*** ***Turing.jl***

REFERENCIAS/BIBLIOGRAFÍA

- K. Team, “¿Qué es la lógica?,” KeepCoding Bootcamps, https://keepcoding.io/blog/que-es-la-programacion-logica/ (accessed Oct. 30, 2023).
- “Programación Lógica,” Wikipe[1] K. Team, “¿Qué es la programación lógica?,” KeepCoding Bootcamps, https://keepcoding.io/blog/que-es-la-programacion-logica/ (accessed Oct. 30, 2023).
- M. Merino, El lenguaje Prolog: un ejemplo del paradigma de programación lógica, https://www.genbeta.com/desarrollo/lenguaje-prolog-ejemplo-paradigma-programacion-logica (accessed Oct. 30, 2023).
- V. Shah, "Logic Programming - A Courseware", 2015. [Online]. Disponible en: https://athena.ecs.csus.edu/~mei/logicp/prolog/programming-examples.html. [Visitado Oct. 30, 2023].
- JavaTpoint, "Prolog Tutorial", 2021. [Online]. Disponible en: https://www.javatpoint.com/prolog. [Visitado Oct. 30, 2023].
- J. Fondren, "Learn X in Y minutes", 2023. [Online]. Disponible en: https://learnxinyminutes.com/docs/mercury/. [Visitado Oct. 30, 2023].
- Universidad de Palermo, UP | Buenos Aires, Argentina. [En línea]. Disponible: https://www.palermo.edu/ingenieria/pdf2014/13/CyT_13_24.pdf
- **Carrasco Vega, C. (2019). Sistema de detección de objetos en movimiento mediante visión artificial. Trabajo Fin de Grado, Universidad Carlos III de Madrid, Madrid, España.**
- **Alonso Amo, Fernando; Villalobos Abarca, Marco. “Programación lógica: un enfoque para desarrollar aplicaciones” [En línea]. Disponible: Accedido el 31 de octubre de 2023. [En línea]. Disponible: https://www.redalyc.org/pdf/944/94401402.pdf**

GRACIAS POR TU ATENCIÓN