PROGRAMACIÓN LÓGICA

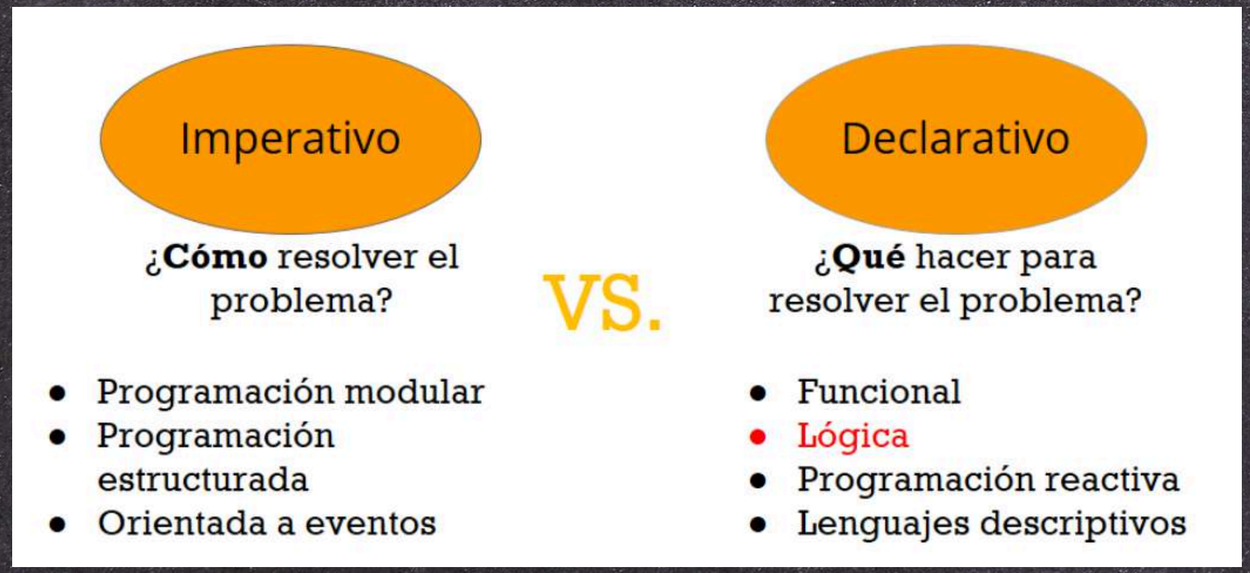
PARADIGMA DE PROGRAMACIÓN

Sebastián Hernández Cerón
Juan Pablo Lozano Aranguren
Luis Fernando Mendez Marques
Santiago Barrera Berrio

ÍNDICE

FILOSOFÍA DEL PARADIGMA

PARADIGMAS DE PROGRAMACIÓN



PROGRAMACIÓN DECLARATIVA

- Abstracción del Control de Flujo
- Transparencia Referencial
- Enfoque en el Qué, No en el Cómo

HISTORIA

RAÍCES TEÓRICAS (1930S–1960S)

1930s-1960s: La lógica de predicados de primer orden y la teoría de autómatas. (Alan Turing, Alonzo Church)

1960: J. Alan Robinson introduce el Principio de Resolución

NACIMIENTO DEL PROLOG (1970S)

1972: Desarrollan Prolog, basado en la unificación y la resolución. (Alain Colmerauer, Robert Kowalski, Philippe Roussel)
1974: Publicación del primer compilador de Prolog

DESARROLLO Y EXPANSIÓN (1980S)

1980s: El interés en la inteligencia artificial (IA) impulsa el desarrollo y la adopción de Prolog.

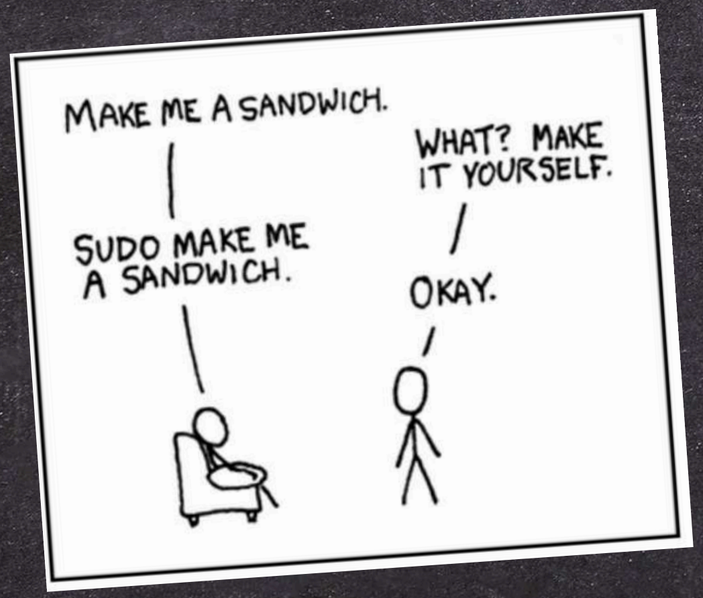
1981: Japón lanza el Proyecto de Computadoras de Quinta Generación (FGCS)

APLICACIONES Y EXTENSIONES (1990S – HOY)

1995-2002: El desarrollo del estándar ISO Prolog

...

PROGRAMACIÓN LÓGICA



Es un paradigma de programación basado en la lógica formal, específicamente en la lógica de predicados de primer orden.

ALGORITMO = LÓGICA + CONTROL

Propuesto por Robert Kowalski

PROGRAMACIÓN LÓGICA

HECHOS

Sentencias que representan datos conocidos.

Juan es padre de María
padre(juan, maria).

REGLAS

Expresiones que relacionan hechos entre sí.

X es abuelo de Y si X es padre de Z y Z es padre de Y
abuelo(X,Y) :- padre(X,Z), padre(Z,Y).

CONSULTAS

Preguntas que se realizan al sistema de lógica para obtener información.

¿Juan es abuelo de María?
?- abuelo(juan, maria).

CARACTERÍSTICAS PRINCIPALES

VARIABLES LÓGICAS

Las variables pueden actuar como variables de entrada y de salida.

?- es_par(4).
?- pares_en_lista([1, 2, 3, 4, 5, 6], L).

INVERSIBILIDAD

Las variables lógicas permiten usar un predicado para extraer información o para validarlo.

?- abuelo(juan, ana).
?- abuelo(juan, Y).

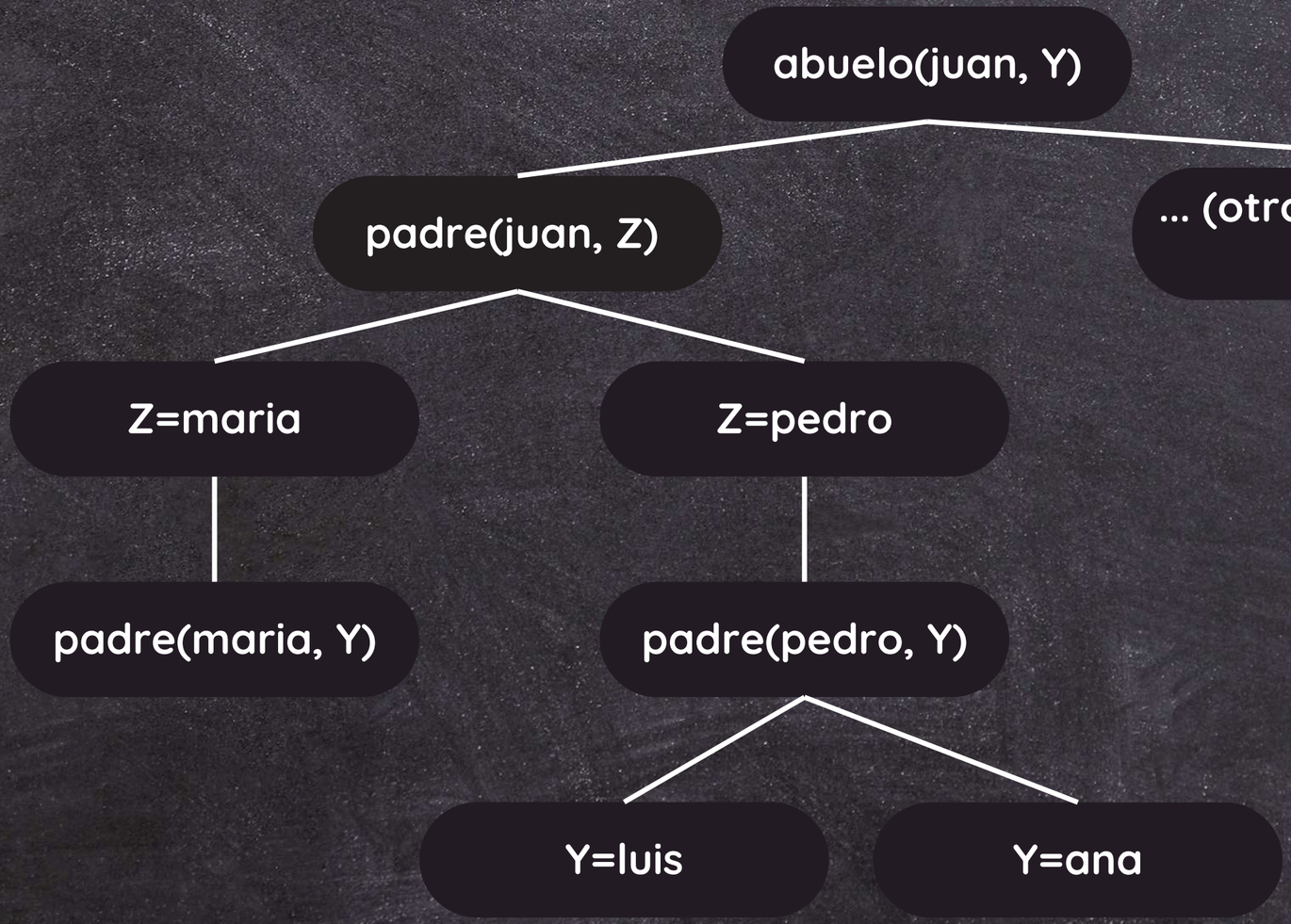
RECURSIVIDAD

No existen los bucles iterativos. Ni for, ni while, ...
Todos los bucles son recursivos

MÚLTIPLES SOLUCIONES (BACKTRACKING)

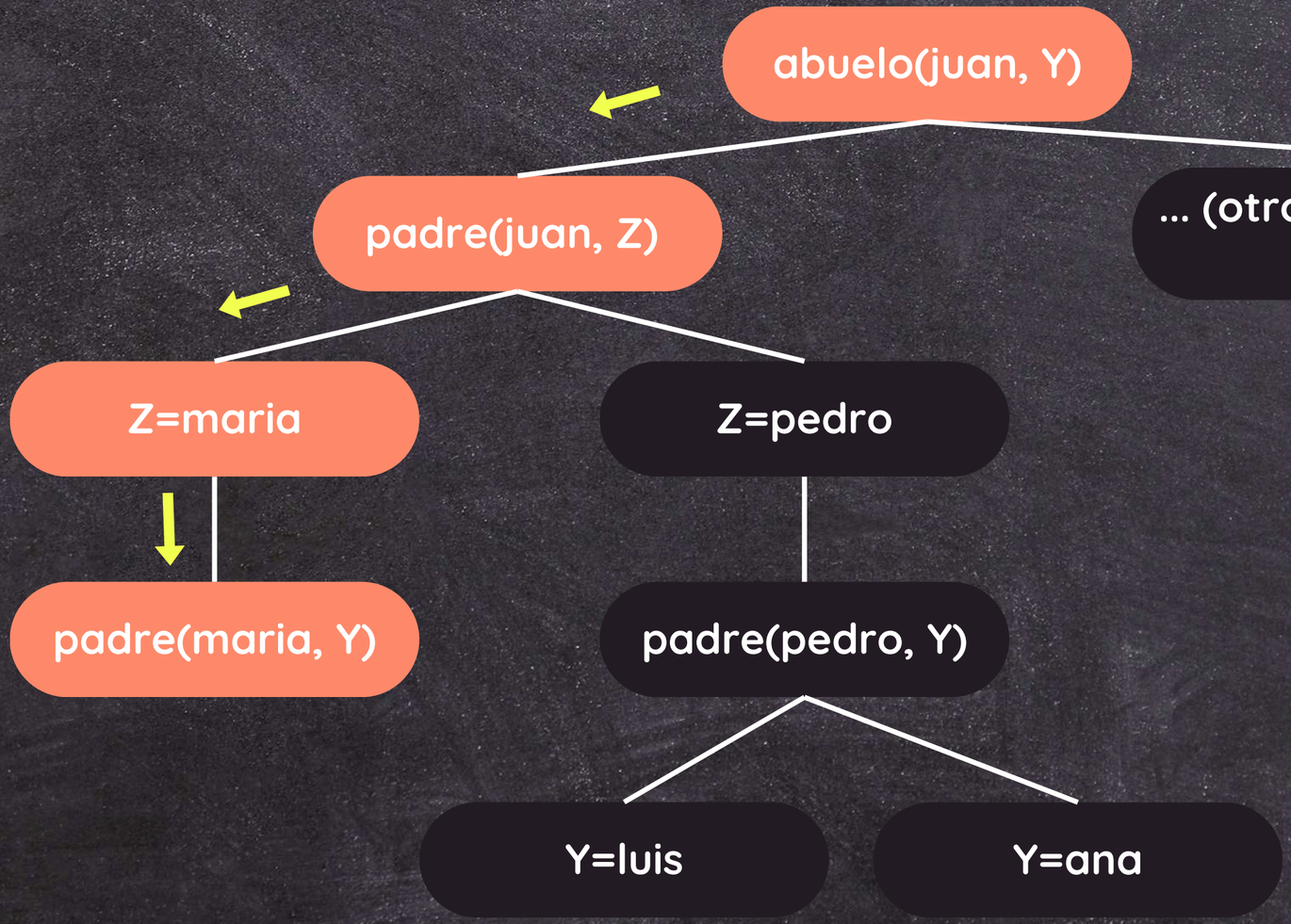
Es posible que la invocación de un predicado devuelva múltiples soluciones, una por cada rama de éxito en el árbol de resolución SLD

EJEMPLO



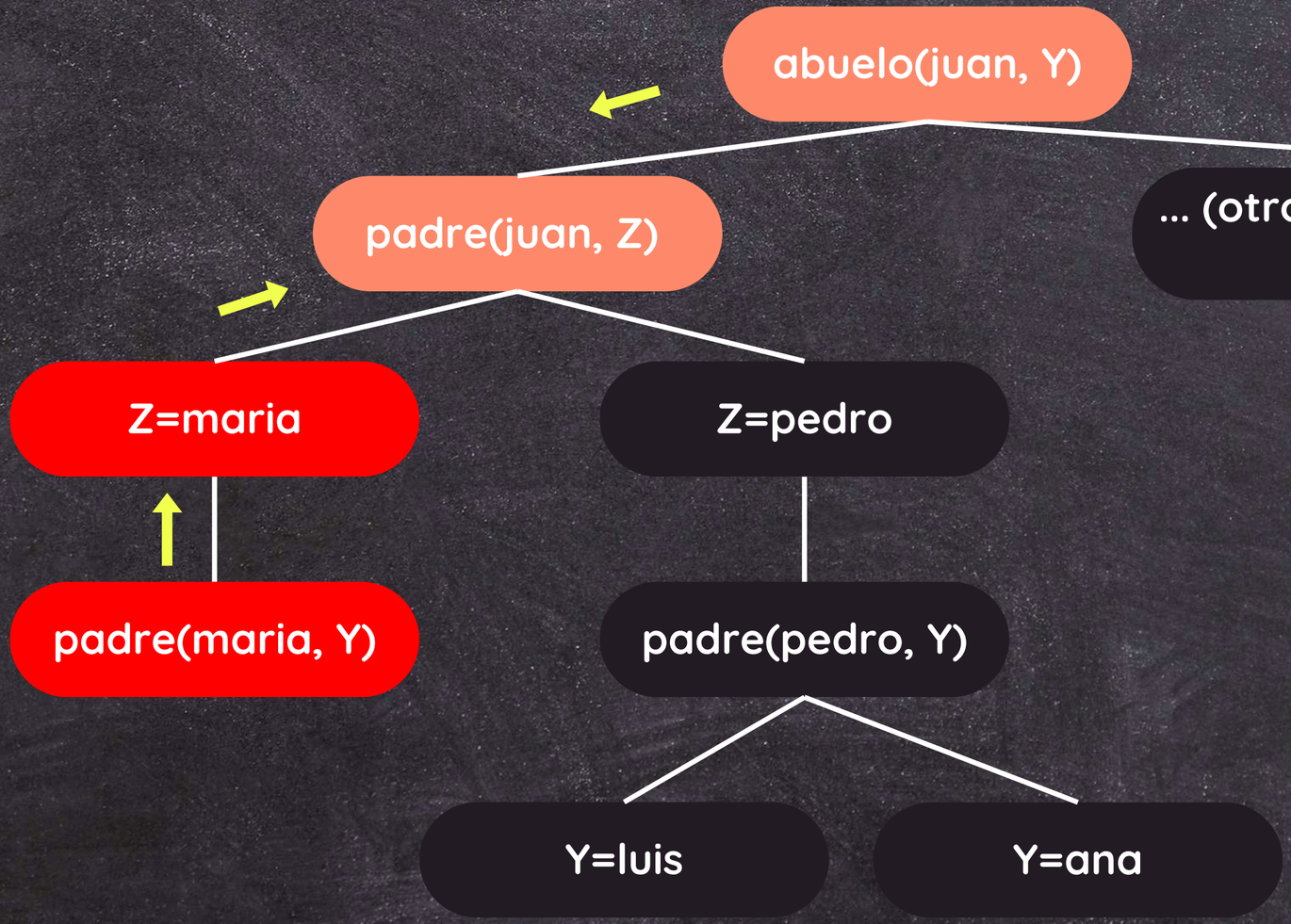

```
% Hechos
    padre(juan, maria).
    padre(juan, pedro).
    padre(pedro, luis).
    padre(pedro, ana).
% Regla
    abuelo(X, Y) :- padre(X, Z),
padre(Z, Y).
% Consulta
    ?- abuelo(juan, Y).
```

EJEMPLO




```
% Hechos
    padre(juan, maria).
    padre(juan, pedro).
    padre(pedro, luis).
    padre(pedro, ana).
% Regla
    abuelo(X, Y) :- padre(X, Z),
padre(Z, Y).
% Consulta
    ?- abuelo(juan, Y).
```

EJEMPLO



% Hechos

```
padre(juan, maria).
padre(juan, pedro).
padre(pedro, luis).
padre(pedro, ana).
```

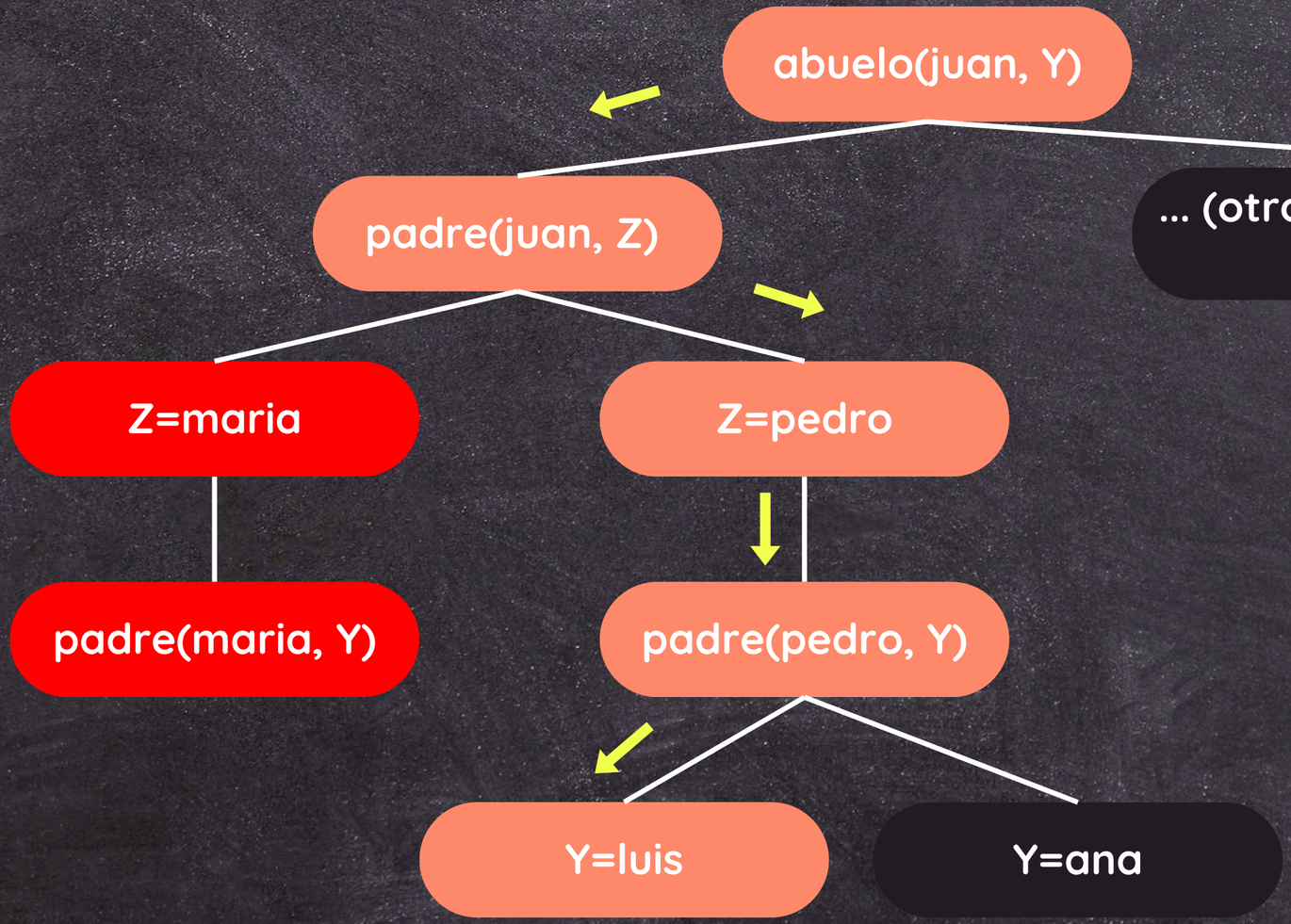
% Regla

```
abuelo(X, Y) :- padre(X, Z), padre(Z, Y).
```

% Consulta

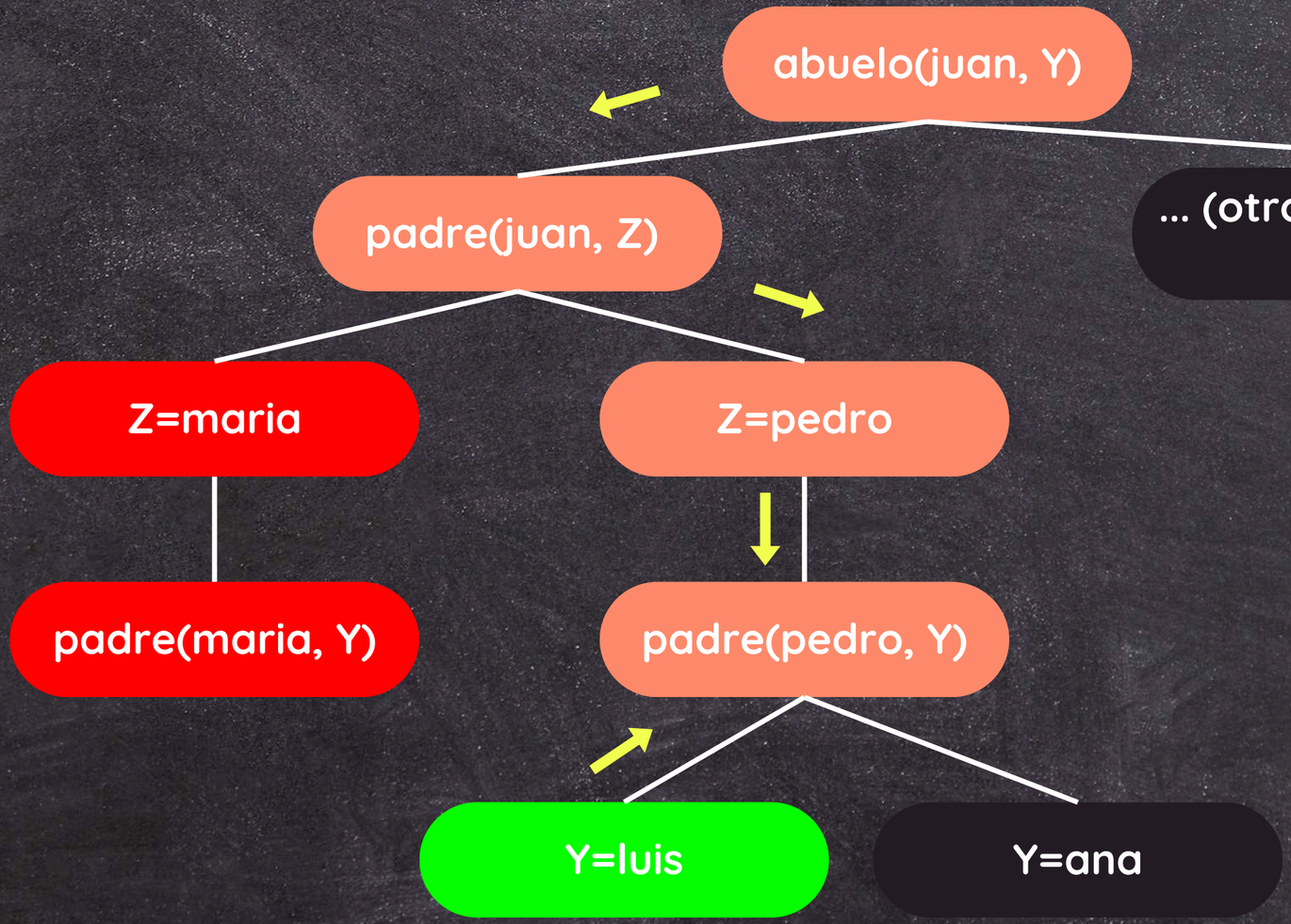
```
?- abuelo(juan, Y).
```

EJEMPLO



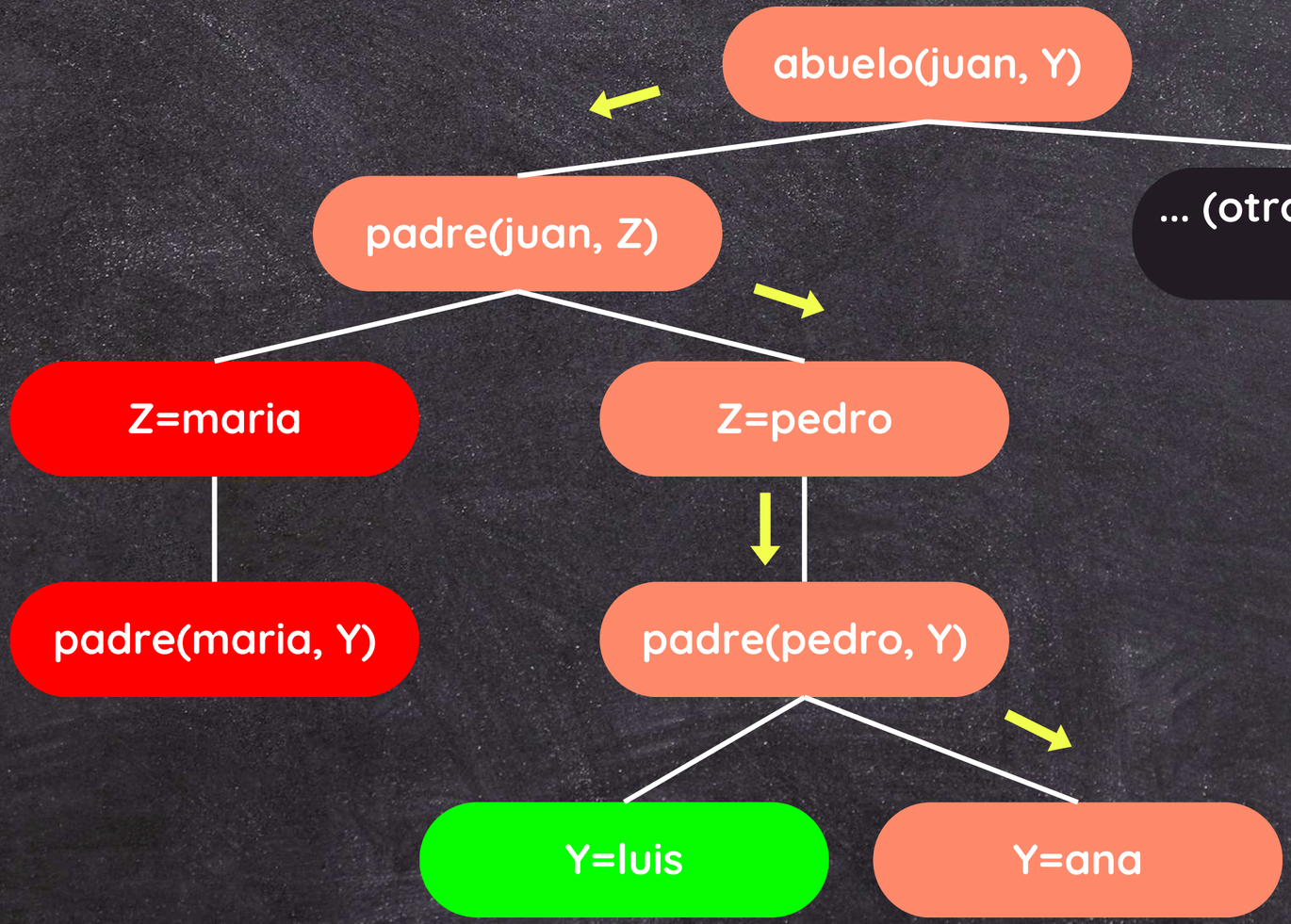

```
% Hechos
    padre(juan, maria).
    padre(juan, pedro).
    padre(pedro, luis).
    padre(pedro, ana).
% Regla
    abuelo(X, Y) :- padre(X, Z),
padre(Z, Y).
% Consulta
     ?- abuelo(juan, Y).
```

EJEMPLO




```
% Hechos
    padre(juan, maria).
    padre(juan, pedro).
    padre(pedro, luis).
    padre(pedro, ana).
% Regla
    abuelo(X, Y) :- padre(X, Z),
padre(Z, Y).
% Consulta
    ?- abuelo(juan, Y).
```

EJEMPLO



% Hechos

```
padre(juan, maria).
padre(juan, pedro).
padre(pedro, luis).
padre(pedro, ana).
```

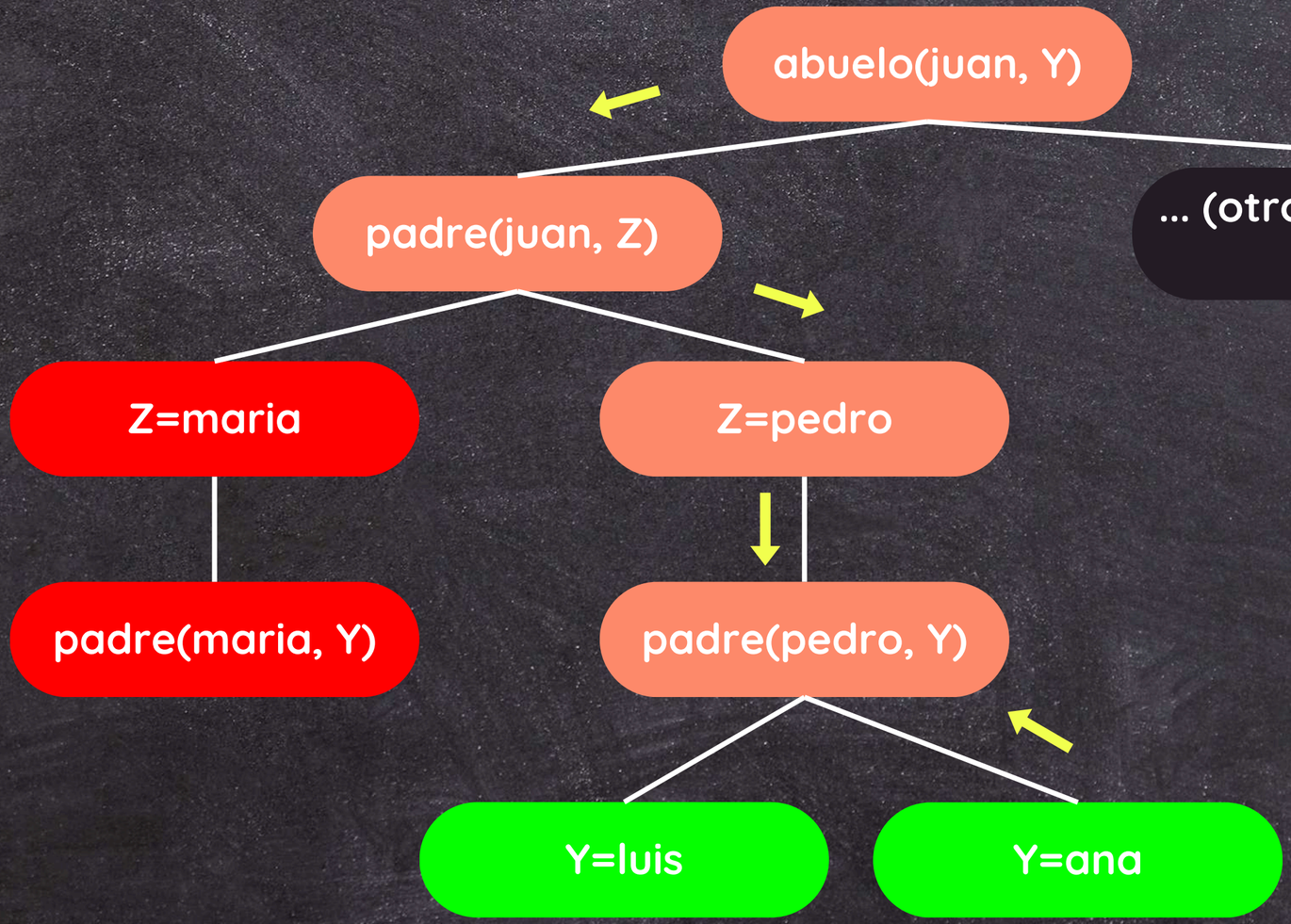
% Regla

```
abuelo(X, Y) :- padre(X, Z), padre(Z, Y).
```

% Consulta

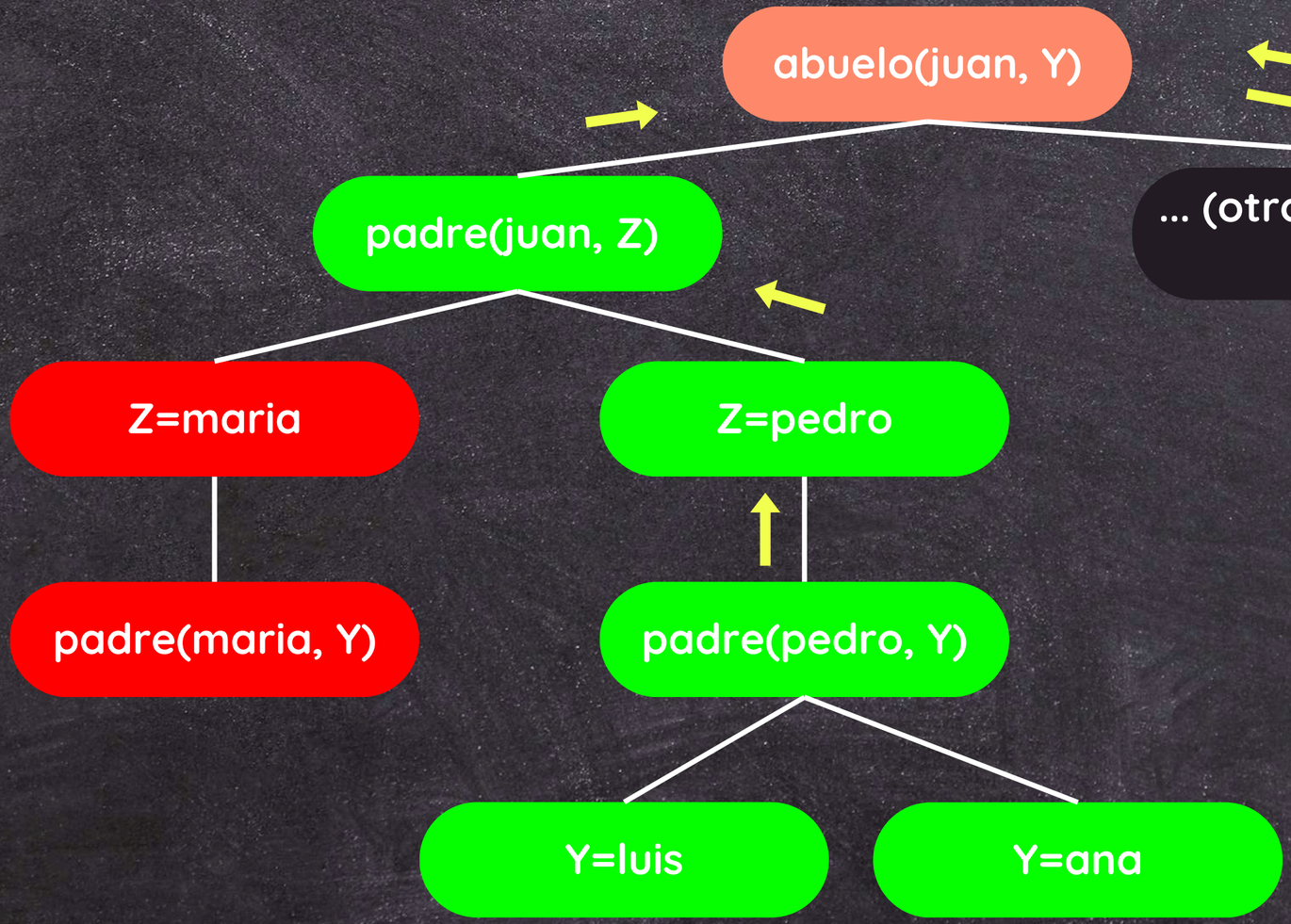
```
?- abuelo(juan, Y).
```

EJEMPLO



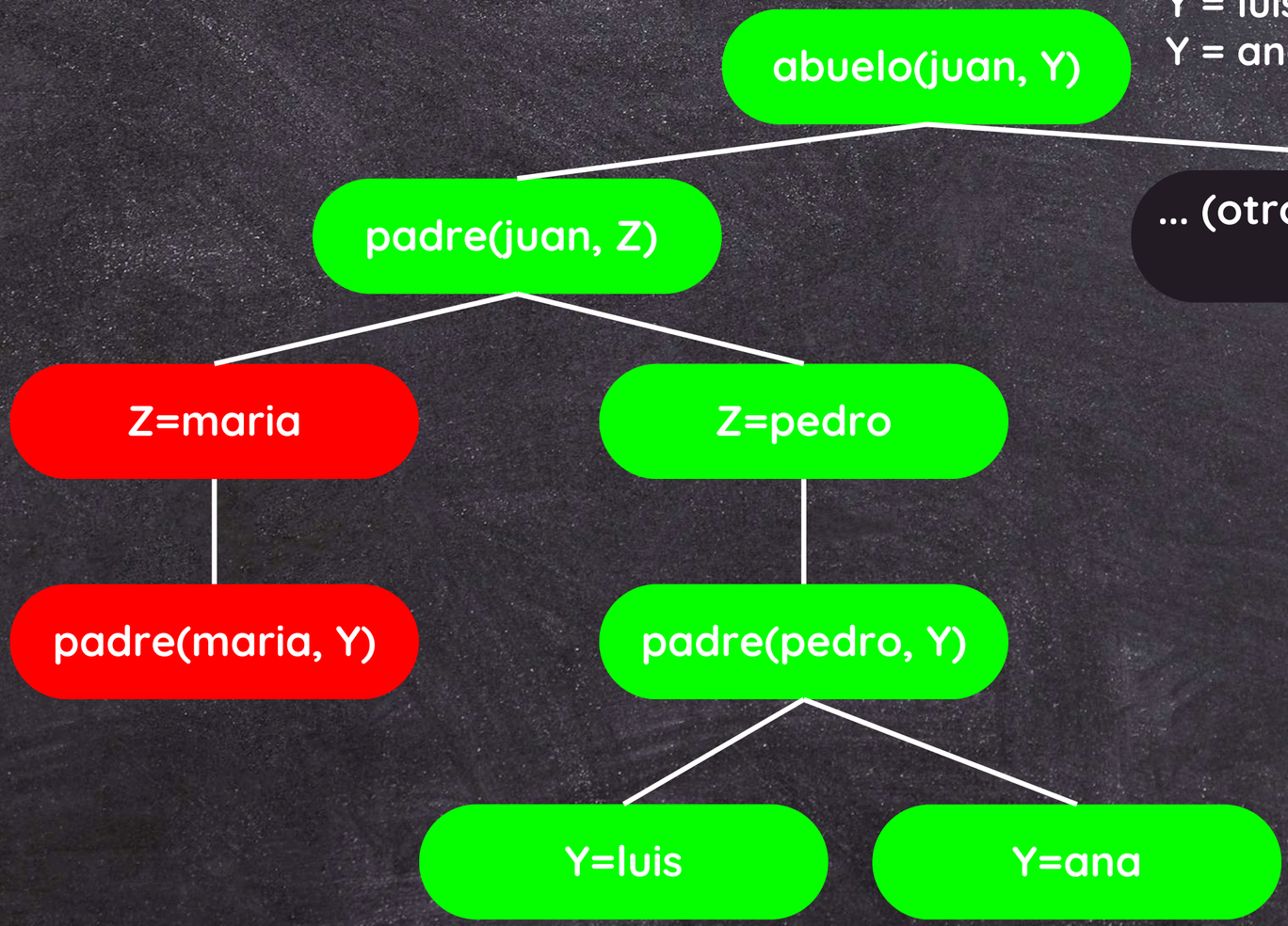

```
% Hechos
    padre(juan, maria).
    padre(juan, pedro).
    padre(pedro, luis).
    padre(pedro, ana).
% Regla
    abuelo(X, Y) :- padre(X, Z),
padre(Z, Y).
% Consulta
    ?- abuelo(juan, Y).
```

EJEMPLO




```
% Hechos
    padre(juan, maria).
    padre(juan, pedro).
    padre(pedro, luis).
    padre(pedro, ana).
% Regla
    abuelo(X, Y) :- padre(X, Z),
padre(Z, Y).
% Consulta
    ?- abuelo(juan, Y).
```

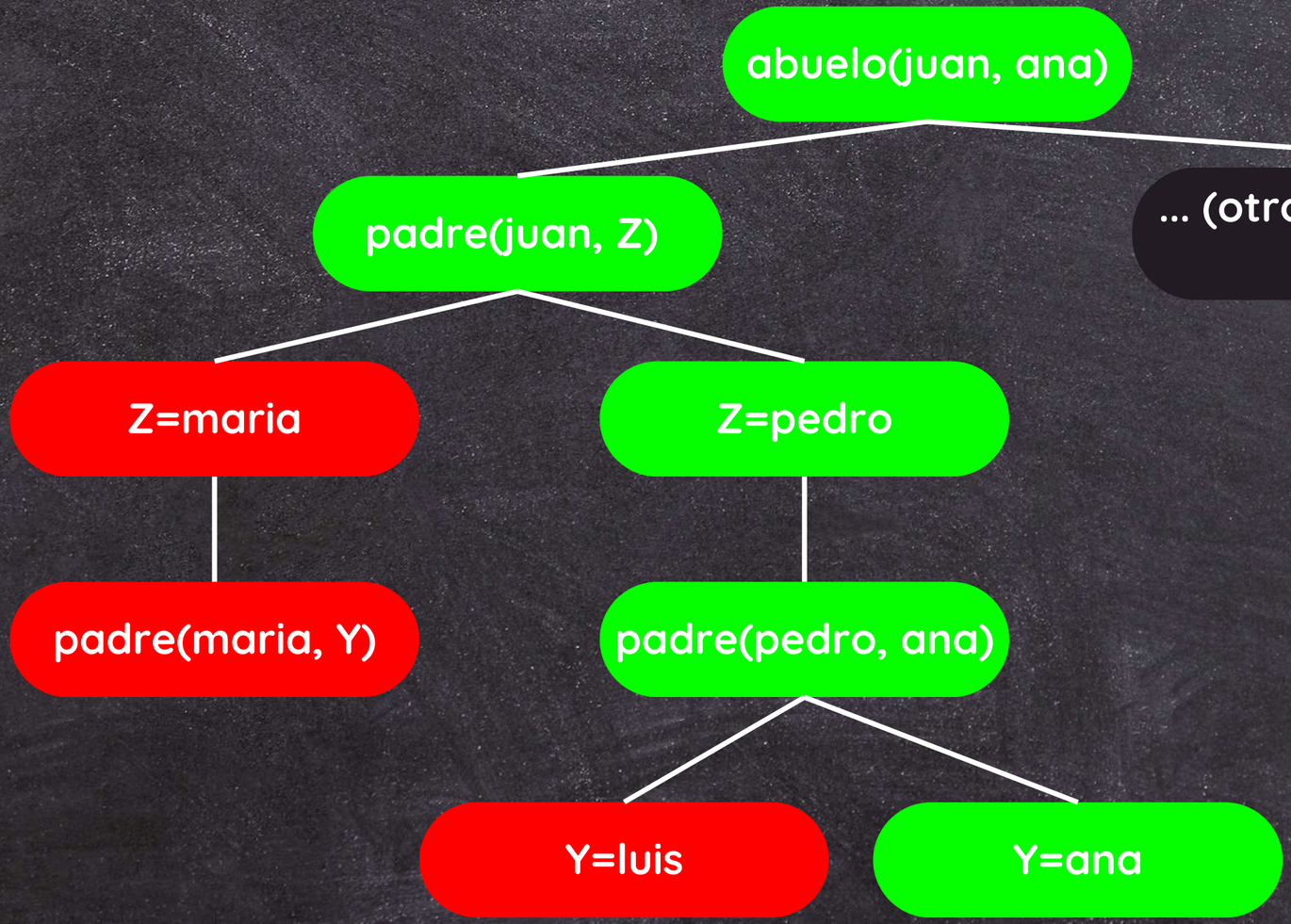
EJEMPLO



Y = luis (nieto de juan a través de pedro)
Y = ana (nieta de juan a través de pedro)

```
% Hechos
    padre(juan, maria).
    padre(juan, pedro).
    padre(pedro, luis).
    padre(pedro, ana).
% Regla
    abuelo(X, Y) :- padre(X, Z),
padre(Z, Y).
% Consulta
    ?- abuelo(juan, Y).
```

EJEMPLO 2




```
% Hechos
    padre(juan, maria).
    padre(juan, pedro).
    padre(pedro, luis).
    padre(pedro, ana).
% Regla
    abuelo(X, Y) :- padre(X, Z),
padre(Z, Y).
% Consulta
    ?- abuelo(juan, ana).
```

TIPO DE CONSULTAS

Consultas de Verificación

Comprueban si ciertos hechos o relaciones son verdaderos según la base de conocimientos.

?- padre(juan, maria).

Consultas de Búsqueda de Variables

Buscan los valores que hacen verdadera una relación dada, resolviendo para las variables implicadas.

?- padre(juan, X).

Consultas de Regla Compleja

Implican múltiples relaciones y reglas, buscando comprobar hechos derivados de reglas más complejas.

abuelo(X, Y) :- padre(X, Z), padre(Z, Y)
?- abuelo(juan, Y).

Consultas con Condiciones

Se pueden hacer consultas con condiciones adicionales para restringir los resultados.

?- padre(X, Y), madre(Y, Z).

Consultas de Existencia

Estas consultas verifican si existe al menos un conjunto de valores que satisfaga una condición.

?- existe(abuelo(juan, Y)).

CONCEPTOS CLAVES

LÓGICA DE PRIMER ORDEN

Alfabeto de la lógica de primer orden

- Los símbolos de L_1 son:

-Constantes individuales:	a, b, c, a_1...	términos
-Variables individuales:	x, y, z, x_1 ...	
-Predicados o relatores:	P, Q, R, R_1 ...	
-Cuantificadores:	$\forall, \exists$	
-Identidad:	$=, \neq$	
-Conectivas:	$\neg, \wedge, \vee, \rightarrow, \leftrightarrow$	
-Auxiliares:	), (,], [	

Es un **sistema formal** diseñado para **estudiar la inferencia** en los lenguajes de primer orden, que son lenguajes formales **que extienden la Lógica Proposicional por medio de cuantificadores y símbolos** que permiten hablar de los objetos de un dominio con predicados y funciones.

Sintaxis: Utiliza símbolos para representar objetos, predicados, conectivas lógicas y cuantificadores

Semántica: Se basa en la interpretación de los símbolos en un universo de discurso, que es el conjunto de objetos sobre los que se razona.

Inferencia: Se refiere a las técnicas para deducir nuevas fórmulas a partir de un conjunto de fórmulas dadas (premisas).

CLÁUSULA DE HORN

Una cláusula de Horn es una disyunción de literales con a lo más un literal positivo. Lleva el nombre del lógico Alfred Horn, quien la dio a conocer en 1951.

$$\neg p \lor \neg q \lor \cdots \lor \neg t \lor u$$

$$(p \land q \land \cdots \land t) \rightarrow u$$

A es mujer y B es padre de A entonces A es hija de B

$$\neg mujer(A) \lor \neg padre(B, A) \lor hija(A, B)$$

$$(mujer(A) \land padre(B, A)) \rightarrow hija(A, B)$$

```
% Hechos
    mujer(a).
    padre(b, a).
% Regla
    hija(X, Y) :- mujer(X), padre(Y, X).
% Consulta
    ?- hija(a, b).
```

RESOLUCIÓN-SLD Y UNIFICACIÓN

Resolución SLD (Resolución lineal con función de selección para cláusulas definidas)

Este método sigue una estrategia de búsqueda en profundidad (depth-first search) y se basa en la resolución de literales para comprobar la veracidad de una consulta.

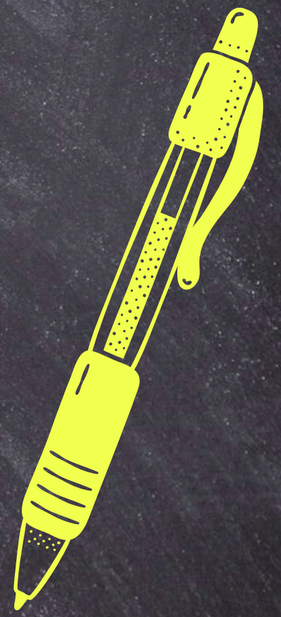
Unificación

La unificación es el proceso de hacer coincidir dos términos, resolviendo sus variables si es necesario.

VENTAJAS Y DESVENTAJAS

VENTAJAS

- Permite describir problemas complejos de manera concisa y natural.
- El motor de inferencia puede deducir automáticamente nueva información a partir de hechos y reglas.
- Es más sencillo modificar un programa declarativo porque las especificaciones están más cerca de la descripción del problema real.



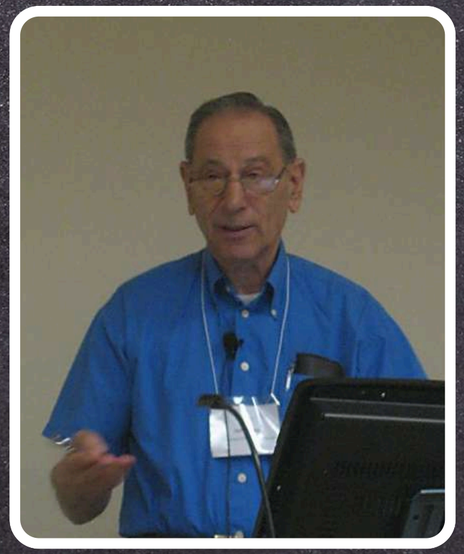
DESVENTAJAS

- Los programas pueden ser lentos para problemas grandes o complejos debido a la búsqueda exhaustiva de soluciones.
- Puede ser difícil rastrear errores debido a la naturaleza declarativa y al proceso de inferencia automática.
- No existen herramientas de depuración efectivas.

LENGUAJES DE PROGRAMACIÓN

DATALOG

70′ - 80′



Jack Minker

- PyDatalog
- Datomic
- Flix
- RecDB

- Unificación
- Evaluación de Reglas
- Fijación de Punto
- Negación Estratificada

```
(<- (:works-for :employee ?x :boss ?y) (:boss :employee-id ?e-id :boss-id ?b-id)
                                       (:employee :id ?e-id :name ?x)
                                       (:employee :id ?b-id :name ?y))
```

ANSWER SET PROGRAMMING

Clingo DLV Smodels

Lógica no monótona

Si $\Gamma \vdash A$, entonces $\Gamma \cup \Delta \vdash A$

Michael Gelfond
Vladimir Lifschitz

Conjuntos Estables

Negación Por Ausencia

Búsqueda de Modelos

Búsqueda Resolución de Conflictos Modelo

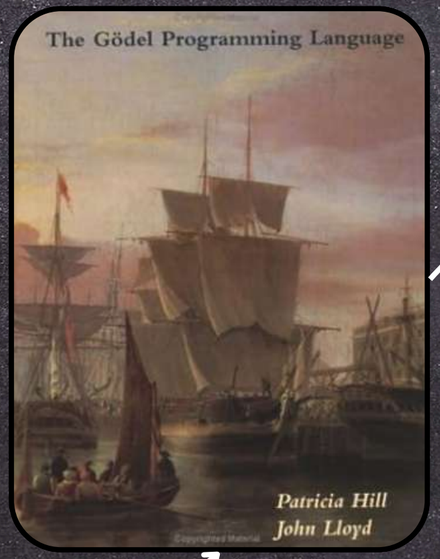
```
%detect possible and suggested places
possiblePlace(Place) :- askFor(TripKind,_), PlaceOffer(Place, TripKind).
suggestPlace(Place) :- possiblePlace(Place), askFor(_,Period),
                       suggestedPeriod(Place, Period),
                       not BadPeriod(Place, Period).

%select packages that the user is possibly interested in
possibleOffer(O) :- TouristicOffer(O, Place), possiblePlace(Place).
```

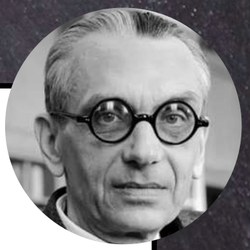
GÖDEL

John Lloyd

Patricia Hill



Gödel recibe su nombre en honor a Kurt Gödel, un renombrado lógico y matemático conocido por sus teoremas de incompletitud.



```
PREDICATE    > : zPz : Integer * Integer;
             < : zPz : Integer * Integer;
             >= : zPz : Integer * Integer;
             =< : zPz : Integer * Integer;
             Interval : Integer * Integer * Integer.
```

Orientado a la Matemática Computacional

Modularidad

Sistema de Tipos Avanzado

Compatibilidad con Lógica de Primer Orden

TWELF

twelf.org/

CURRY

curry.pages.ps.informatik.uni-kiel.de/curry-lang.org/

MERCURY

mercurylang.org/

EJEMPLO

PROLOG

Sudoku Solver

En este código, definimos las reglas y restricciones para que el Sudoku sea resuelto automáticamente.

HECHOS

valid([]): Una lista vacía es válida.

```
:- use_module(library(clpfd)).

valid([]).
valid([Head|Tail]) :-
    all_different(Head),
    valid(Tail).
```

% Definición de la validez de una lista de listas.

valid([Head|Tail]): Una lista es válida si su cabeza contiene elementos diferentes y el resto de la lista también es válido.

REGLAS

```
sudoku(Puzzle, Solution) :-
    Solution = Puzzle,
    Puzzle = [S11, S12, S13, S14, S15, S16, S17, S18, S19,
              S21, S22, S23, S24, S25, S26, S27, S28, S29,
              S31, S32, S33, S34, S35, S36, S37, S38, S39,
              S41, S42, S43, S44, S45, S46, S47, S48, S49,
              S51, S52, S53, S54, S55, S56, S57, S58, S59,
              S61, S62, S63, S64, S65, S66, S67, S68, S69,
              S71, S72, S73, S74, S75, S76, S77, S78, S79,
              S81, S82, S83, S84, S85, S86, S87, S88, S89,
              S91, S92, S93, S94, S95, S96, S97, S98, S99],
    Solution ins 1..9,
```

% Regla principal para resolver el Sudoku.

Definir Filas y Columnas y Cuadrados 3x3 del Sudoku

```
Row1 = [S11, S12, S13, S14, S15, S16, S17, S18, S19],
Row2 = [S21, S22, S23, S24, S25, S26, S27, S28, S29],
Row3 = [S31, S32, S33, S34, S35, S36, S37, S38, S39],
Row4 = [S41, S42, S43, S44, S45, S46, S47, S48, S49],
Row5 = [S51, S52, S53, S54, S55, S56, S57, S58, S59],
Row6 = [S61, S62, S63, S64, S65, S66, S67, S68, S69],
Row7 = [S71, S72, S73, S74, S75, S76, S77, S78, S79],
Row8 = [S81, S82, S83, S84, S85, S86, S87, S88, S89],
Row9 = [S91, S92, S93, S94, S95, S96, S97, S98, S99],
```

Filas

```
Col1 = [S11, S21, S31, S41, S51, S61, S71, S81, S91],
Col2 = [S12, S22, S32, S42, S52, S62, S72, S82, S92],
Col3 = [S13, S23, S33, S43, S53, S63, S73, S83, S93],
Col4 = [S14, S24, S34, S44, S54, S64, S74, S84, S94],
Col5 = [S15, S25, S35, S45, S55, S65, S75, S85, S95],
Col6 = [S16, S26, S36, S46, S56, S66, S76, S86, S96],
Col7 = [S17, S27, S37, S47, S57, S67, S77, S87, S97],
Col8 = [S18, S28, S38, S48, S58, S68, S78, S88, S98],
Col9 = [S19, S29, S39, S49, S59, S69, S79, S89, S99],
```

Columnas

```
Square1 = [S11, S12, S13, S21, S22, S23, S31, S32, S33],
Square2 = [S14, S15, S16, S24, S25, S26, S34, S35, S36],
Square3 = [S17, S18, S19, S27, S28, S29, S37, S38, S39],
Square4 = [S41, S42, S43, S51, S52, S53, S61, S62, S63],
Square5 = [S44, S45, S46, S54, S55, S56, S64, S65, S66],
Square6 = [S47, S48, S49, S57, S58, S59, S67, S68, S69],
Square7 = [S71, S72, S73, S81, S82, S83, S91, S92, S93],
Square8 = [S74, S75, S76, S84, S85, S86, S94, S95, S96],
Square9 = [S77, S78, S79, S87, S88, S89, S97, S98, S99],
```

Cuadrados 3x3

APLICAR RESTRICCIONES DE VALIDEZ

```
valid([Row1, Row2, Row3, Row4, Row5, Row6, Row7, Row8, Row9]),
valid([Col1, Col2, Col3, Col4, Col5, Col6, Col7, Col8, Col9]),
valid([Square1, Square2, Square3, Square4, Square5, Square6, Square7, Square8, Square9]).
```

CONSULTA

```
?- sudoku([5, 3, _, _, 7, _, _, _, _,
           6, _, _, 1, 9, 5, _, _, _,
           _, 9, 8, _, _, _, _, 6, _,
           8, _, _, _, 6, _, _, _, 3,
           4, _, _, 8, _, 3, _, _, 1,
           7, _, _, _, 2, _, _, _, 6,
           _, 6, _, _, _, _, 2, 8, _,
           _, _, _, 4, 1, 9, _, _, 5,
           _, _, _, _, 8, _, _, 7, 9], Solution).
```

?- sudoku(Puzzle, Solution): Para resolver un Sudoku, se proporciona el puzzle inicial y se obtiene la solución.

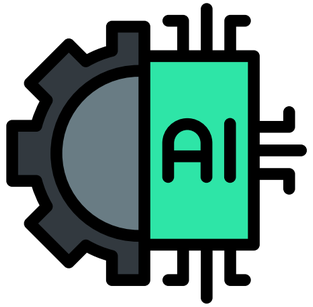
SOLUCIÓN

```
Solution =
[5, 3, 4, 6, 7, 8, 9, 1, 2, 6, 7, 2, 1, 9, 5, 3, 4, 8, 1, 9, 8, 3, 4, 2, 5, 6, 7, 8, 5, 9, 7, 6, 1, 4, 2, 3, 4, 2, 6, 8, 5, 3, 7, 9, 1, 7, 1, 3, 9, 2, 4, 8, 5, 6, 9, 6, 1, 5, 3, 7, 2, 8, 4, 2, 8, 7, 4, 1, 9, 6, 3, 5, 3, 4, 5, 2, 8, 6, 1, 7, 9]
```

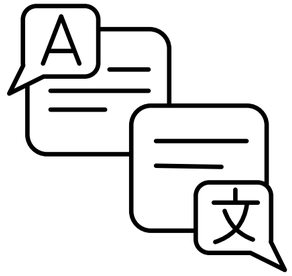
5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

APLICACIONES

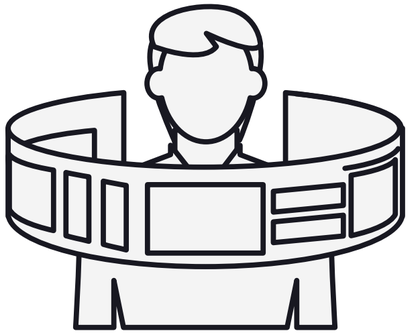
SISTEMAS EXPERTOS



PROCESAMIENTO DEL LENGUAJE NATURAL (NLP)



SISTEMAS DE CONTROL



SISTEMAS EXPERTOS

¿QUÉ SON?

Aquellos que emulan la toma de desiciones de un humano, con base a su experiencia en el dominio específico de un tema.

¿CÓMO TRABAJAN?

Utilizan una base de conocimientos formada por hechos y reglas lógicas para ofrecer recomendaciones, diagnósticos o soluciones.

FUNCIONAMIENTO

Entrada de datos

Aplicación de reglas

Deducción de conclusiones

Presentación de resultados

CASOS

- Diagnóstico médico
  - MYCIN: (1970) Diagnostico de infecciones bacterianas y recomendar tratamientos antibióticos.
  - DENDRAL: Identificación de estructuras moleculares a partir de datos espectrométricos.
- Asesoría legal

PROCESAMIENTO DEL LENGUAJE NATURAL (NLP)

ANÁLISIS LINGÜÍSTICO

Analisis de estructuras lingüísticas, como oraciones y párrafos. Se pueden definir reglas lógicas para descomponer y comprender la gramática de las frases.

IA
Analizadores de texto

FORMALISMOS METAGRAMATICALES

Sistemas formales que describen la estructura gramatical de un idioma., es decir, permiten expresar generalizaciones sobre la sintaxis de los lenguajes naturales de manera concisa.

Oración activa (sujeto + verbo + objeto)
Oración pasiva (objeto + verbo + “ser” + participio pasado).

TRATAMIENTO LÓGICO DE PROBLEMAS LINGÜÍSTICOS

Se pueden abordar problemas lingüísticos desde una perspectiva lógica. Por ejemplo, se pueden expresar reglas para resolver ambigüedades en la interpretación de oraciones.

REPRESENTACIÓN DEL SIGNIFICADO

Se utiliza para representar el significado de las expresiones lingüísticas. Esto puede incluir la construcción de ontologías o redes semánticas.

SISTEMAS DE CONTROL

MODELADO DEL ESPACIO DE ESTADOS

Representa todas las posibles configuraciones de un sistema y se describe mediante estados y transiciones. Se modela empleando reglas y hechos.

- Estados: Representan situaciones específicas en el entorno del agente.
- Acciones: Describen las posibles transiciones de un estado a otro.
- Restricciones: Condiciones que deben cumplirse para que una acción sea válida.

VENTAJAS

- Declaratividad: Especifica "qué" se desea lograr en lugar de "cómo" hacerlo.
- Flexibilidad: Facilita la modificación y extensión de reglas y restricciones sin reescribir todo el código.
- Optimización: Utiliza técnicas de búsqueda y resolución de restricciones para encontrar soluciones óptimas de manera eficiente.

DESVENTAJAS

- Escalabilidad: A medida que el espacio de estados crece, la complejidad computacional puede volverse prohibitiva.
- Captura de Conocimientos: Definir todas las reglas y restricciones relevantes, puede ser laborioso y requiere conocimiento experto.
- Ambigüedad e Incertidumbre: Manejar incertidumbres en el entorno y en la ejecución de acciones puede ser complejo.

MUCHAS GRACIAS

REFERENCIAS

- Acosta, Isabel. (s.f.). Lógica de Primer Orden. Recuperado de https://slideplayer.es/slide/4796321/
- Universidad de Sevilla. (2023). Sintaxis y Semántica de la Lógica de Primer Orden. Recuperado de https://www.cs.us.es/cursos/liti-2023/SintaxisSemanticaLPO.md.html
- Universidad Nacional de Colombia. (2024). Presentaciones sobre Lógica y Teoría (2016-2 y 2019-2). Recuperado de https://ferestrepoca.github.io/paradigmas-de-programacion/proglogica/logica_teoria/presentaciones.html
- Galiana,Silva. Francesc, Josep (2019). El paradigma de programación lógico. [Video]. YouTube. https://www.youtube.com/watch?v=vQX0R-J1YSo
- DW Español. Inteligencia artificial: ¿una nueva era para la cirugía? (13 de enero de 2024). Accedido el 27 de mayo de 2024. [Video en línea]. Disponible: https://www.youtube.com/watch?v=TnGC9M5UYDk
- Deutsche Welle. “Colombia: Resuelven primer caso con ayuda de robot ChatGPT – DW – 02/02/2023”. dw.com. Accedido el 27 de mayo de 2024. [En línea]. Disponible: https://www.dw.com/es/resuelven-en-colombia-el-primer-caso-jurídico-con-la-ayuda-de-robot-chatgpt/a-64597510

REFERENCIAS

- Castillo, A. F. (2004). Sistemas expertos. Monografias.com. https://www.monografias.com/trabajos16/sistemas-expertos/sistemas-expertos
- A. Chavez. “sistemas expertos”. SlideShare. Accedido el 27 de mayo de 2024. [En línea]. Disponible: https://es.slideshare.net/uni_fcys_sistemas/sistemas-expertos-14737155
- Sánchez, M. C., & Meléndez, S. M. (2000). SISTEMAS EXPERTOS. Universidad Nacional Mayor de San Marcos. https://sistemas.unmsm.edu.pe/
- Alty, J. L. (1983). Sistemas expertos. Paraninfo.
- Willison, D. J., & Kennedy, I. M. (1999). Guillain-Barre syndrome after heat stroke London. Mar 1999. Journal of Neurology, Neurosurgery and Psychiatry, 66(3), 339-340.

REFERENCIAS

- “Sistemas Expertos en Inteligencia Artificial: tipos, usos y ventajas”. Ciberseguridad. Accedido el 27 de mayo de 2024. [En línea]. Disponible: https://ciberseguridad.com/guias/nuevas-tecnologias/inteligencia-artificial/sistemas-expertos/
- “Mycin - Wikipedia”. Wikipedia, the free encyclopedia. Accedido el 27 de mayo de 2024. [En línea]. Disponible: https://en.wikipedia.org/wiki/Mycin
- “Dendral - Wikipedia”. Wikipedia, the free encyclopedia. Accedido el 27 de mayo de 2024. [En línea]. Disponible: https://en.wikipedia.org/wiki/Dendral
- S. Badaró, L. J. Ibañez y M. J. Agüero, Sistemas Expertos: Fundamentos, Metodologías y Aplicaciones. Buenos Aires: Universis Palermo, 2013.

REFERENCIAS

- Russell, S. J., & Norvig, P. (2010). Artificial intelligence: A modern approach (3rd ed.). Upper Saddle River, NJ: Pearson Prentice Hall. https://www.pearson.com/en-us/search.html?aq=Russell%20Artificial-Intelligence-A-Modern-Approach-3rd-Edition
- Poole, D. L. (2016). Logic in practice (5th ed.). Boston, MA: Pearson. https://www.routledge.com/Logic-in-Practice/Stebbing/p/book/9780367426309
- Nilsson, N. J. (1980). Principles of artificial intelligence. Palo Alto, CA: Tioga Press. https://link.springer.com/book/9783540113409
- Russell, S. J., & Norvig, P. (2003). Artificial intelligence: A modern approach (2nd ed.). Upper Saddle River, NJ: Pearson Prentice Hall. https://www.pearson.com/en-us/subject-catalog/p/artificial-intelligence-a-modern-approach/P200000003500/9780137505135
- Stone, P. (2007). Introduction to artificial intelligence (2nd ed.). Upper Saddle River, NJ: Pearson Prentice Hall. https://www.pearson.com/en-us/subject-catalog/p/artificial-intelligence-a-modern-approach/P200000003500/9780137505135
- Russell, S. J., & Norvig, P. (2010). Artificial intelligence: A modern approach (3rd ed.). Upper Saddle River, NJ: Pearson Prentice Hall. https://www.pearson.com/en-us/search.html?aq=Russell%20Artificial-Intelligence-A-Modern-Approach-3rd-Edition
- Kaelbling, L. P., & Littman, M. L. (2006). Reinforcement learning: A survey. Journal of Artificial Intelligence Research, 27, 237-285. https://www.jair.org/index.php/jair/article/view/10166
- Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. Cambridge, MA: MIT Press. https://mitpress.mit.edu/9780262039246/reinforcement-learning/