PROGRAMACIÓN LÓGICA

Kevin Andres Castro Garcia

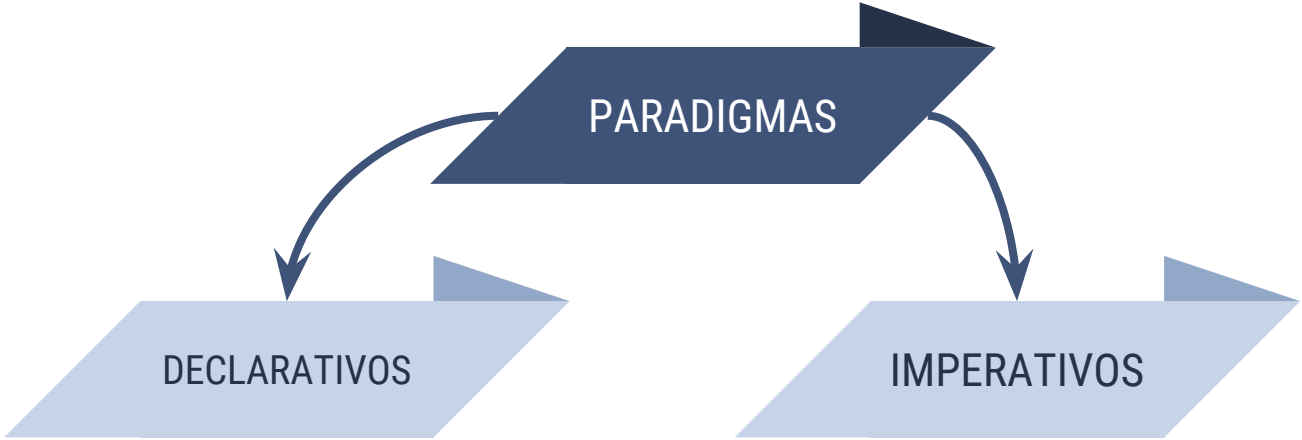
Juan David Rojas Sanchez

CONTENIDO

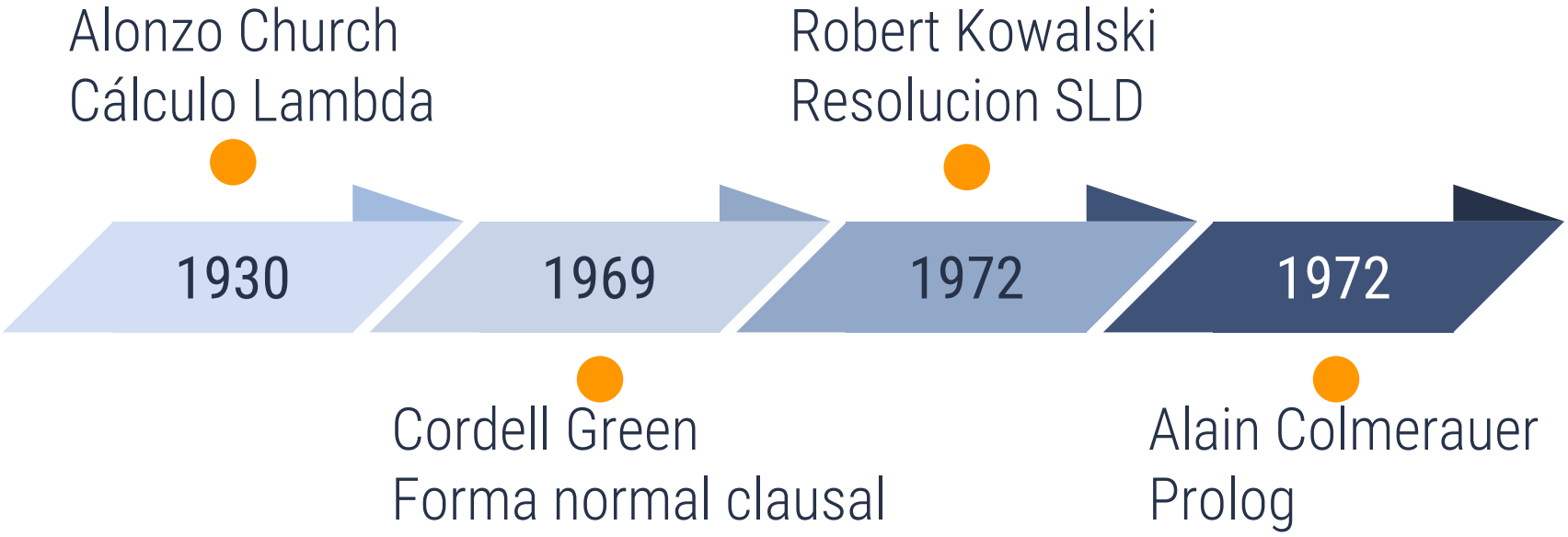
1

INTRODUCCIÓN

PARADIGMAS DE PROGRAMACIÓN



HISTORIA

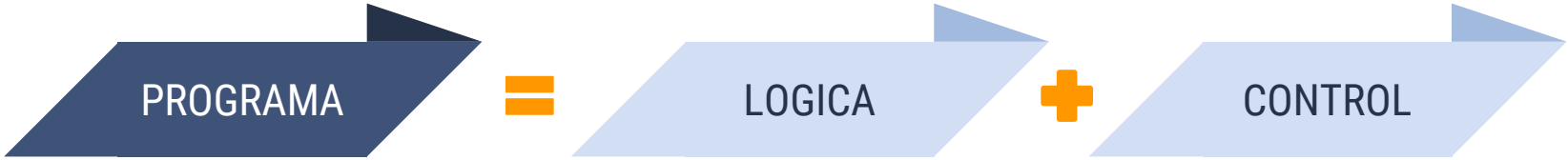

Alonzo Church
Cálculo Lambda
1930
Cordell Green
Forma normal clausal
1969
Robert Kowalski
Resolucion SLD
1972
Alain Colmerauer
Prolog
1972

2

FILOSOFÍA DEL PARADIGMA

Aplicación de las reglas de la lógica para inferir conclusiones a partir de datos.

PROGRAMACION LOGICA



- **Lógica:** Hechos y reglas para representar conocimiento.
- **Control:** Deducción lógica para dar respuestas.
- Resolución de problemas de forma automática.

3

CONCEPTOS CLAVE

CONCEPTOS DE LÓGICA

- **Proposición:** Sentencia lógica que puede ser verdadera o falsa; atomica o compuesta.
- **Lógica simbólica**: Proposiciones, relaciones entre estas, inferencia.
- **Cálculo de predicados:** Tipo de lógica simbólica usado en programación lógica.

CÁLCULO DE PREDICADOS

- También llamado lógica de primer orden, diseñado para estudiar inferencia.
- Poder expresivo muy superior al de la lógica proposicional.
- Permite uso de variables y funciones.

CÁLCULO DE PREDICADOS - ELEMENTOS

- **Constantes:** Símbolos que representan un objeto conocido.
- **Variables:** Símbolos que representan diferentes objetos en diferentes tiempos.
- **Predicados:** Funciones que pueden ser verdadero o falso.

CÁLCULO DE PREDICADOS - OPERADORES

NOMBRE	SIMBOLO
Negacion	$\neg$
Conjunción	$\wedge$
Disyunción	$\vee$
Implicación	$\Rightarrow$

NOMBRE	SIMBOLO
Equivalencia	$\Leftrightarrow$
Universal	$\forall$
Existencial	$\exists$

CÁLCULO DE PREDICADOS - EJEMPLO

- "Para todo conjunto x existe un conjunto y, tal que la cardinalidad de y es mayor que la cardinalidad de x".

$$(\forall x)\{CONJ(x) \Rightarrow (\exists y)(\exists u)(\exists v)[CONJ(y) \wedge CARD(x,u) \wedge CARD(y,v) \wedge MAY(v,u)]\}$$

CLÁUSULAS DE HORN

- Secuencia de literales que contiene a lo sumo uno de sus literales positivos (disyunción de literales).

$$\neg p \vee \neg q \vee \ldots \vee \neg t \vee u$$

$$(p \wedge q \wedge \ldots \wedge t) \rightarrow u$$

- **Cláusula determinada:** Cláusula de HORN con exactamente un literal positivo.
- **Cláusula objetivo**(consulta):Sin ningún literal positivo.

CLÁUSULAS DE HORN - EJEMPLO(Prolog)

```
1 % Hechos:
2 es_padre(A,B).
3 es_padre(B,C).
4
5 % Reglas:
6 es_abuelo(A,C):- es_padre(A,B),es_padre(B,C).
```

- **Lectura formal: A** es abuelo de **C** si **A** es padre de **B** y **B** es padre de **C**.
- **Lectura de Predicados:**

(es_padre(A,B) ∧ es_padre(B,C)) ➡ es_abuelo(A,C)

RESOLUCION SLD

$(p \land q \land \ldots \land t) \Rightarrow u$
$P \Rightarrow Q \equiv \neg P \lor Q$

- **RESOLUCION**

$$(\textbf{man}(X) \Rightarrow \textbf{mortal}(X)) \land \textbf{man}(\text{socrates})$$

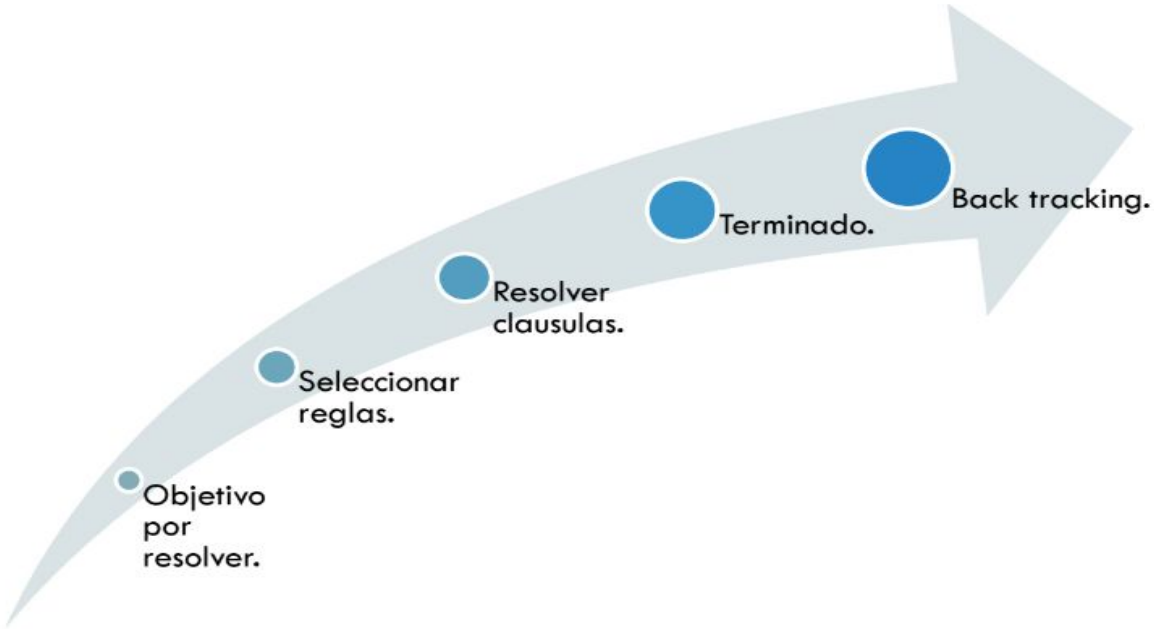
$$(\textbf{man}(\text{socrates}) \Rightarrow \textbf{mortal}(\text{socrates})) \land \textbf{man}(\text{socrates})$$

$$(\neg\,\textbf{man}(\text{socrates}) \lor \textbf{mortal}(\text{socrates})) \land \textbf{man}(\text{socrates})$$

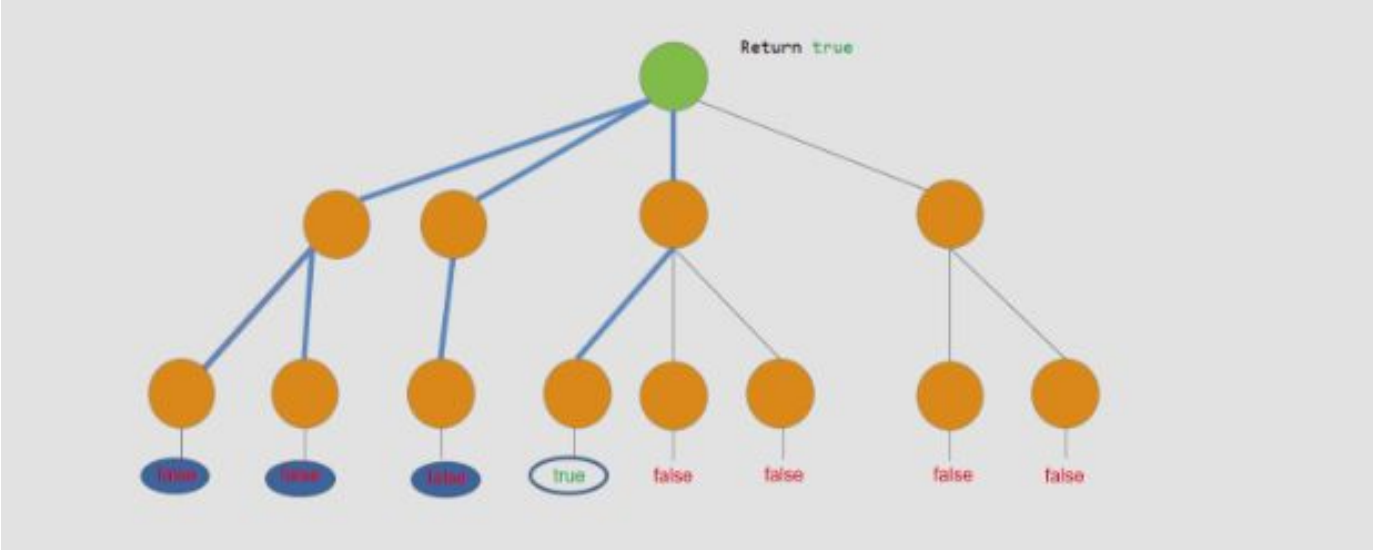
$$((\neg\,\textbf{man}(\text{socrates}) \land \textbf{man}(\text{socrates})) \lor (\textbf{mortal}(\text{socrates}) \land \textbf{man}(\text{socrates})))$$

$$(\textbf{mortal}(\text{socrates}) \land \textbf{man}(\text{socrates}))$$

RESOLUCION SLD (Selective Linear Definite clause)

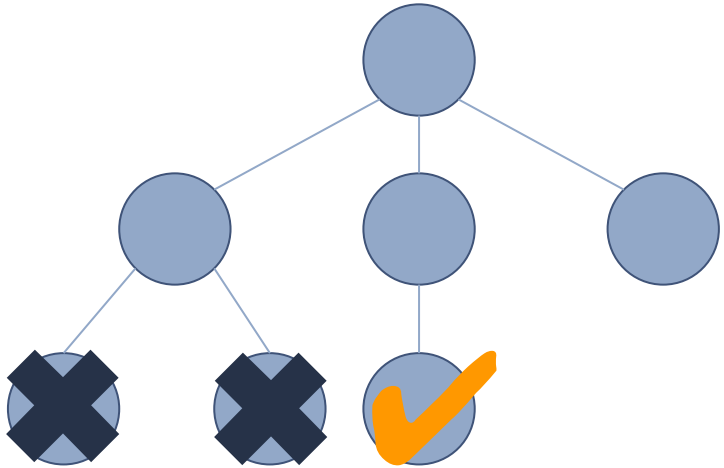


RESOLUCIÓN SLD



BACKTRACKING

- Cuando un objetivo tiene sub-objetivos se usa backtracking.
- Si no se logra probar verdadero con un sub-objetivo se devuelve y escoge otro.



CONCEPTOS DE IMPLEMENTACIÓN

- **Hecho:** Declaración, cláusula o proposición cierta o falsa, el hecho establece una relación entre objetos y es la forma más sencilla de sentencia.

```
1.  humano(Socrates).
2.  % Socrates es humano
3.  parent(Juan, María).
4.  % Juan es el padre de María
```

CONCEPTOS DE IMPLEMENTACIÓN

- **Regla:** Implicación o inferencia lógica que deduce nuevo conocimiento, la regla permite definir nuevas relaciones a partir de otras ya existentes.

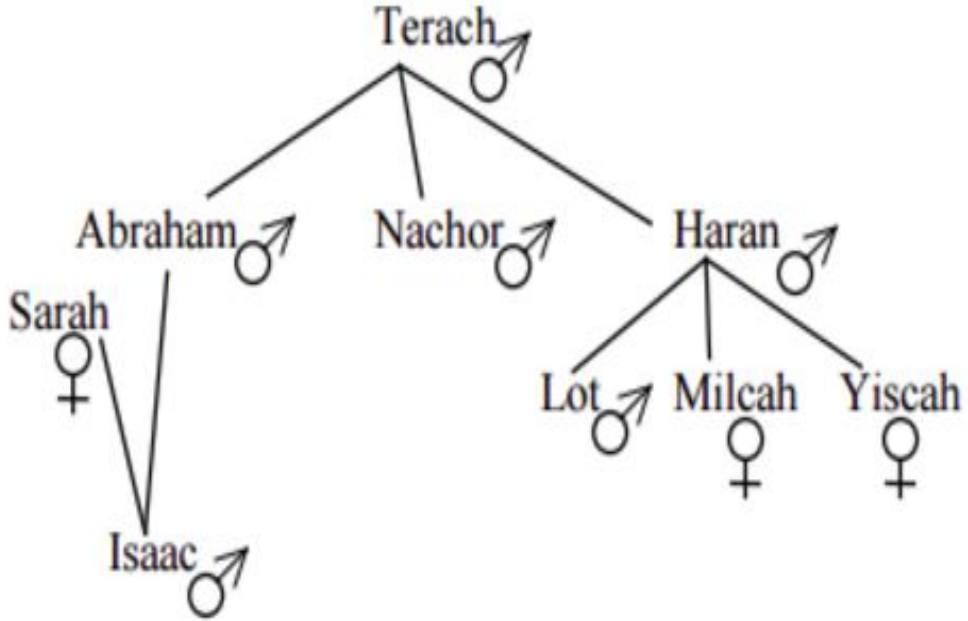
```
1.  mortal (x):- humano(x).
2.  % x es mortal si x es humano
3.  grandparent(A, B):- parent(A, X), parent(X, B).
4.  % A es el abuelo de B si A es el
5.  % padre de X y X es el padre de B
```

CONCEPTOS DE IMPLEMENTACIÓN

- **Consulta:** Se especifica el problema, objetivo, o proposición a demostrar.

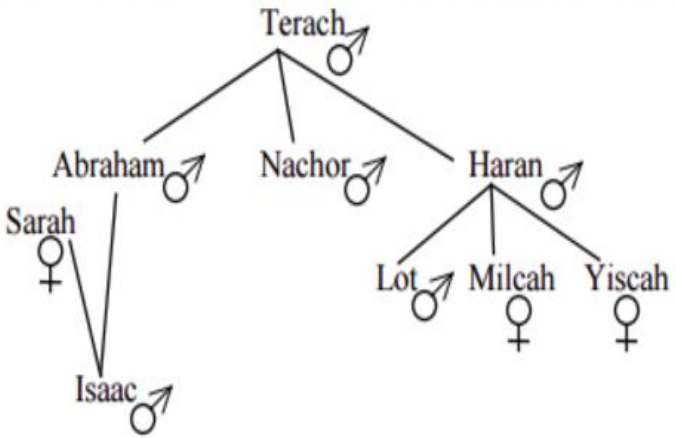
```
1.  mortal(X):- humano(X).
2.  humano(Socrates).
3.  ?- mortal(Socrates).
4.  true.
```

Ejemplo



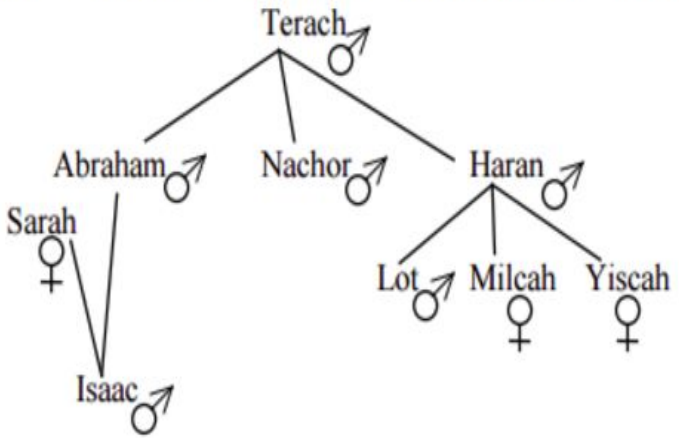
HECHOS

```
es_padre(terach, abraham).
es_padre(terach, nachor).
es_padre(terach, haran).
es_padre(abraham, isaac).
es_padre(haran, lot).
es_padre(haran, milcah).
es_padre(haran, yiscah).

es_madre(sarah, isaac).

es_hombre(terach).
es_hombre(abraham).
es_hombre(nachor).
es_hombre(haran).
es_hombre(isaac).
es_hombre(lot).

es_mujer(sarah).
es_mujer(milcah).
es_mujer(yiscah).
```



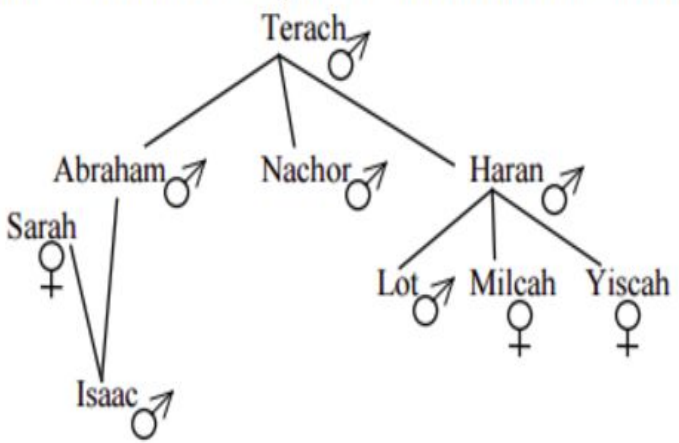
REGLAS

```
es_hijo(X,Y):- es_padre(Y,X), es_hombre(X).

es_hija(X,Y):- es_padre(Y,X), es_mujer(X).

es_abuelo(X,Z):- es_padre(X,Y), es_padre(Y,Z).
```



CONSULTAS

```
2 ?- es_padre(haran,lot), es_hombre(lot).
true .

3 ?- es_padre(abraham,lot), es_hombre(lot).
false.

4 ?- es_padre(abraham,X), es_hombre(X).
X = isaac.

5 ?- es_padre(haran,X), es_mujer(X).
X = milcah ;
X = yiscah.

6 ?- es_hijo(lot,haran).
true .

7 ?- es_hija(X,haran).
X = milcah ;
X = yiscah.
```



RECURSIVIDAD

- **Problema:** Hallar el abuelo de una persona.

```
1.   parent(david, john).
2.   parent(jim, david).
3.   parent(steve, jim).
4.   parent(nathan, steve).
5.   grandparent(A, B):- parent(A, X), parent(X, B).
```

- **Problema:** Hallar los ancestros de una persona.

```
1.   ancestor(A,B) :- parent(A, B).
2.   ancestor(A,B) :- parent(A, X), parent(X, B).
3.   ancestor(A,B) :- parent(A, X), parent(X, Y), parent(Y, B).
4.   ancestor(A,B) :- parent(A, X), parent(X, Y), parent(Y, Z), parent(Z,B).
```

RECURSIVIDAD

- **Solución:** Recursividad.

```
1.   ancestor(A, B) :- parent(A, B).
2.   ancestor(A, B) :- parent(A, X), ancestor(X, B).
3.
4.   ?- ancestor(X, john).
```

```
1.   X = david
2.   X = jim
3.   X = steve
4.   X = nathan
```

4

VENTAJAS Y DESVENTAJAS

VENTAJAS

- Fácil de escribir programas sin conocer bien el lenguaje.
- Cercanía a las especificaciones del problema realizadas con lenguajes formales.
- Se puede modificar el componente de control sin modificar la lógica del algoritmo.

DESVENTAJAS

- No existen herramientas de depuración efectivas.
- Pocas áreas de aplicación, es poco utilizado en problemas reales.
- Los motores de inferencia son limitados.
- Retorna false en caso de no tener suficiente información para cierta consulta

5

LENGUAJES

LENGUAJES

PROLOG

- Primer lenguaje.
- Mas conocido y usado.

GÖDEL

- Polimorfismo.
- Tipos de datos.
- Módulos.
- Meta-programación.

LENGUAJES

ALF

- Algebraic Logic Functional.
- Une funcional y lógica.

MERCURY

- Alto nivel.
- Mundo real.
- Compilado.
- Traducción a C.

6

EJEMPLOS

Fibonacci:

```
1.  fibonacci(0,0).
2.  fibonacci(1,1).
3.  fibonacci(N,X) :-
4.       N > 1,
5.       N1 is N-1,
6.       fibonacci(N1,X1),
7.       N2 is N-2,
8.       fibonacci(N2, X2),
9.       X is X1+X2.
10. ?- fibonacci(10,X)
---------------------------
1.  X=55
```

Naive sort:

```
1.  naive_sort(List,Sorted) :-
2.       perm(List,Sorted),
3.       is_sorted(Sorted).
4.  is_sorted([ ]).
5.  is_sorted([_]).
6.  is_sorted([X,Y|T]):-
7.       X=<Y,
8.       is_sorted([Y|T]).
9.  ?- naive_sort([3,2,1],X)
```

```
1.  X = [1, 2, 3]
```

Maximo Comun Divisor:

```
 1.   MODULE GCD.
 2.   IMPORT Integers.
 3.   PREDICATE Gcd : Integer * Integer * Integer.
 4.   Gcd(i,j,d) <-
 5.         CommonDivisor(i,j,d) &
 6.         ~SOME [e] (CommonDivisor(i,j,e) & e > d).
 7.   PREDICATE CommonDivisor : Integer * Integer * Integer.
 8.   CommonDivisor(i,j,d) <-
 9.         IF (i = 0 \/ j = 0)
10.         THEN
11.               d = Max(Abs(i),Abs(j))
12.         ELSE
13.               1 =< d =< Min(Abs(i),Abs(j)) &
14.               i Mod d = 0 &
15.               j Mod d = 0.
```

MERCURY

Fibonacci:

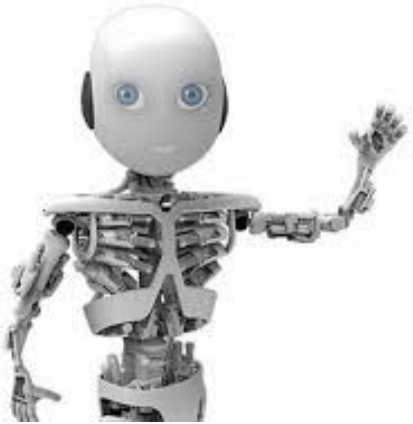
```
 1.   :- module fib.
 2.   :- interface.
 3.   :- import_module io.
 4.   :- pred main(io::di, io::uo) is det.
 5.
 6.   :- implementation.
 7.   :- import_module int.
 8.
 9.   :- func fib(int) = int.
10.   fib(N) = (if N =< 2 then 1 else fib(N - 1) + fib(N - 2)).
11.   main(!IO) :-
12.           io.write_string("fib(10) = ", !IO),
13.           io.write_int(fib(10), !IO),
14.           io.nl(!IO).
```

7

APLICACIONES

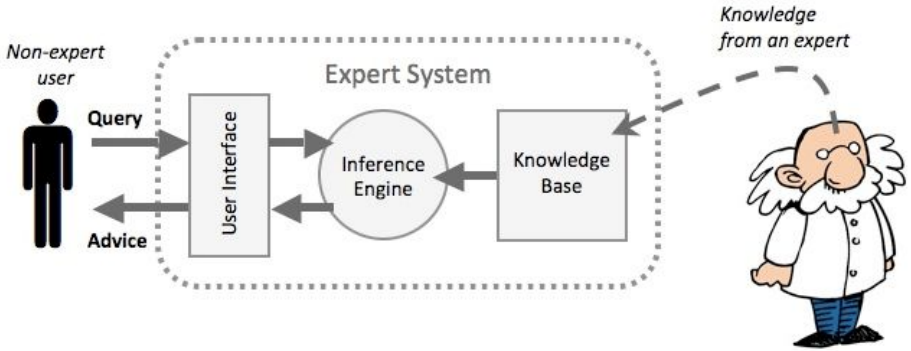
INTELIGENCIA ARTIFICIAL

Este es el uso por excelencia de la programación lógica, se puede encontrar la mejor respuesta para un juego.



SISTEMAS EXPERTOS

Son sistemas informáticos que simulan el proceso de aprendizaje, de memorización, de razonamiento, de comunicación y de acción en consecuencia de un experto en cualquier rama de la ciencia.



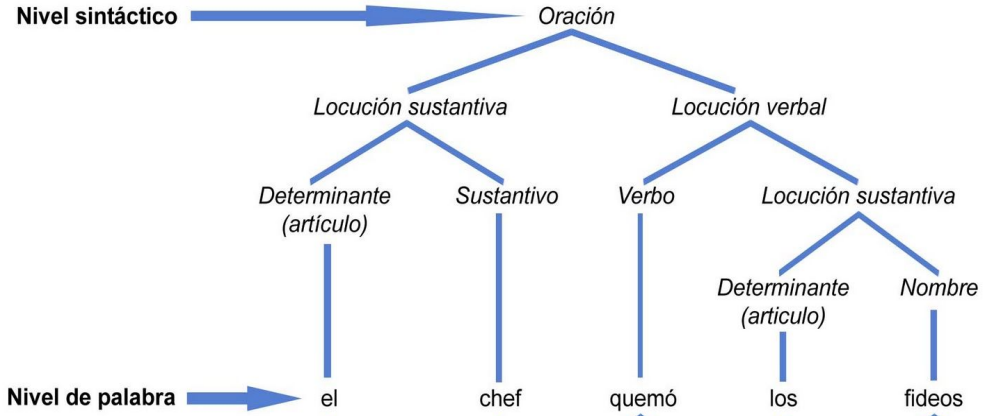
ANÁLISIS DE LENGUAJE NATURAL

Procesamiento del lenguaje natural, donde un programa es capaz de comprender (con limitaciones) la información contenida en una expresión lingüística humana.

1.

Nivel del directorio → *Enunciado: Quemar (chef, fideos)* *Deducción: el chef es incompetente*

.



OTRAS APLICACIONES

- Prueba de teoremas de forma automática.
- Bases de conocimientos.
- Resolución de problemas.

8

REFERENCIAS

1. http://inteligenciaartificialbelarmino.blogspot.com.co/2011/04/introduccion-la-programacion-logica.html
2. http://dit.upm.es/~gfer/ssii/rcsi/rcsisu59.html
3. http://blog.koalite.com/2013/08/que-es-la-programacion-logica/
4. https://en.wikipedia.org/wiki/Logic_programming
5. https://www.cs.cmu.edu/~fp/courses/lp/lectures/lp-all.pdf
6. https://en.wikibooks.org/wiki/Prolog/Recursive_Rules

7. http://nbviewer.jupyter.org/url/ferestrepoca.github.io/paradigmas-de-programacion/proglogica/logica_teoria/Programacion-logica.ipynb
8. http://www.it.uc3m.es/jvillena/irc/practicas/estudios/Lenguajes_Logicos.pdf
9. http://www.cs.us.es/~jalonso/pub/2006-ej_prog_declarativa.pdf
10. https://www.youtube.com/watch?v=b9lg9zUY2eY
11. http://www.academia.edu/7698501/APLICACI%C3%93N_DE_PROGRAMACI%C3%93N_L%C3%93GICA_EN_DETECCI%C3%93N_DE_SINTAXIS_EN_EL_LENGUAJE_NATURAL

GRACIAS