

PROLOG: Es un lenguaje declarativo, tipificando Hechos, Reglas de inferencia y Objetivo, sin indicar Cómo se obtiene el término (o regras o los datos primarios).

Una resolución S(D) (en CH & DFS, m=0-C, Toma 1 literal de cada Cláusula. Limpie DFS+DEF).

↳ Cláusulas de Definición: ≤ 1 POS, ≠ NEG.

Cláusulas OBJETIVAS: 0 POS, ≠ NEG.

En mundo cerrado, los significa que lo que no se declara explícitamente es falso.

El término tipo muere por los términos.

Los programar de Prolog tienen formato por Hechos y Reglas de inferencia. Entonces programar la suma realizando consultas sobre dicha base.

ZOMBIE(JUAN).

ZOMBIE(VALENIA).

TOMO-MATE-DESPUES(JUAN, CARLOS).

TOMO-MATE-DESPUES(LARA, JUAN).

INFECTADO(ERNESTO).

INFECTADO(X):- ZOMBIE(X).

INFECTADO(X):- ZOMBIE(Y), TOMO-MATE-DESPUES(Y,X).

¿ Está infectado Ernesto?

QUERY: ? - INFECTADO(ERNESTO)

Vemos qué regla unifica:

1 INFECTADO(ERNESTO). ✓ → TOME → Encuentra Regla.

2 ✓ NEDO

3 INFECTADO(X): ZOMBIE(X)

5) INFECTADO(X):- ZOMBIE(X).

X = ENNESTO

ZOMBIE(ENNESTO)

↳ ZOMBIE(ENNESTO) ? ZOMBIE(JUAN) X

5

↳ ZOMBIE(ENNESTO) ? ZOMBIE(VALENA) X

FAIL \rightsquigarrow La coma me llevó 4

NEGO

6) INFECTADO(Y):- ZOMBIE(Y), TONO-MATE-DESPUES(Y,X).

7

Y = VALENA

Y = JUAN

TONO-MATE-DESPUES(VALENA, ENNESTO)

↳ TONO-MATE-DESPUES(VALENA, ENNESTO) ? TONO-MATE-DESPUES(JUAN, CARLOS) X

↳ TONO-MATE-DESPUES(VALENA, ENNESTO) ? TONO-MATE-DESPUES(CUNA, JUAN) X

FAIL 10

TONO-MATE-DESPUES(JUAN, ENNESTO)

↳ TONO-MATE-DESPUES(JUAN, ENNESTO) ? TONO-MATE-DESPUES(JUAN, CARLOS) X

↳ TONO-MATE-DESPUES(JUAN, ENNESTO) ? TONO-MATE-DESPUES(CUNA, JUAN) X

FAIL 9

11 ~) FALSE. No queda mate para explorar

SINTÁXIS:

· VARIABLES: Van en mayúsculas. No volverán que todavía no fueron ligados. Después de ligarse ya no pueden ser modificadas.

\rightsquigarrow X

· NÚMEROS: 10, 15.6

· ÁTOMOS: Empiezan con minúscula o tienen algún Comillón simple.

CLARA

- TÉRMINOS COMPLEJOS: Combinación de Atómicos seguidos de un Argumento, Cada uno de los cuales es un Término.
 - TOMO_MATE :- DESPUES(CLARA, JUAN)

- TÉRMINO: VARIABLE, NÚMERO, ATÓMICO O TÉRMINO COMPLICADO.

- CLÁUSULA: Línea del programa, termina con punto. Puede ser un Hecho o una regla.

- HECHO: ZOMBIE(JUAN).

- REGLA: INFECTADO(X) :- ZOMBIE(Y), TOMO_MATE_DESPUES(Y, X).

Define una relación entre PREDICADOS.

?
≤

- PREDICADO: Colección de cláusulas.

$$\hookrightarrow \underbrace{Q(S(x), Y, S(z))}_{\text{CLÁUSULA}} :- Q(X, Y, Z) \equiv \left\{ \underbrace{Q(S(X), Y, S(Z)),}_{\text{ATÓMICOS}} \underbrace{\neg Q(X, Y, Z)}_{\text{CLÁUSULA}} \right\}$$

RDD: FORMA CLÁUSULA, CONJUNCIÓN DE DISJUNCIÓN DE LITERALES:

OJO: PREDICADO PROLOG ≠ PREDICADO RESOLUCIÓN

↪ Colección de cláusulas

↪ Solo es un PREDICADO si se le ponen literales en un literal.

- OBJETIVO / GOAL: Es el predicado que se cumple al final de PROLOG.

↪ ? INFECTADO(X).

HECHO ?
NATURAL(CERO). ≡ {NATURAL(CERO)}

↪ Término Atómico
↪ Término Compuesto

NATURAL(SUC(X)) :- NATURAL(X). ≡ PREDICADO {NATURAL(SUC(X)), LITERAL Término }
CLÁUSULA

Definir:

- MAYOR_2(X) que es verdadero cuando X es mayor que 2.

Por lo tanto cualquier NATURAL de la forma SUC(SUC(SUC(X))) es verdadera, podemos definirlo como hecho y que aparece en la segunda cláusula.

MAYOR(A(X)) :- NATURAL(SUC(SUC(SUC(X)))).

$\downarrow \{x_1 = SUC(SUC(x))\}$ $(NATURAL(SUC(x_1)) :- NATURAL(x_1))$

NATURAL(SUC(SUC(x)))

$\downarrow \{x'_1 = SUC(x)\}$ $(NATURAL(SUC(x'_1)) :- NATURAL(x'_1))$

NATURAL(SUC(x))

$\downarrow \{x''_1 = x\}$ $(NATURAL(SUC(x''_1)) :- NATURAL(x''_1))$

NATURAL(x)

NATURAL(x).

$\downarrow \{x'''_1 = CERO\}$

NATURAL(CERO).

4

Ora las frases obligan a que para que sea Mayor que Dos, tengas MENOS que 3.

Entonces $SUC(SUC(SUC(SUC(ZERO))))$ ✓

$SUC(SUC(SUC(SUC(SUC(ZERO)))))$ ✓ ...

Tú entiendes como genera "una o sol".

DEFINICIÓN : ESPAR

ESPAR(CERO).

ESPAR(SUC(SUC(x))) :- ESPAR(x).

Método 2

Algo parecido en CONDOS en V, qué forma debe tener. El impar no me importa.

DEFINICIÓN : MENOR

Si $x > A, B \wedge \forall v \exists A < B \leq v \wedge A \rightarrow v$ si A llega a CERO \rightarrow el menor es el menor $SUC(CERO)$

MENOR(CERO, SUC(x)).

MENOR(SUC(x), SUC(y)) :- MENOR(x, y).

¿Qué sucede si lo compro...?

MENOR(CERO, UNO) : FALSE, NO UNIFICADA. ✓

MENOR(CERO, X)

$\downarrow x = SUC(x')$

MENOR(CERO, SUC(x')) \rightarrow TRUE.

MENOR(SUC(SUC(CERO)), CERO).

NEON(x) \cup NOUNIFICA \cup SUC(y)

GATO(GANFIELD).

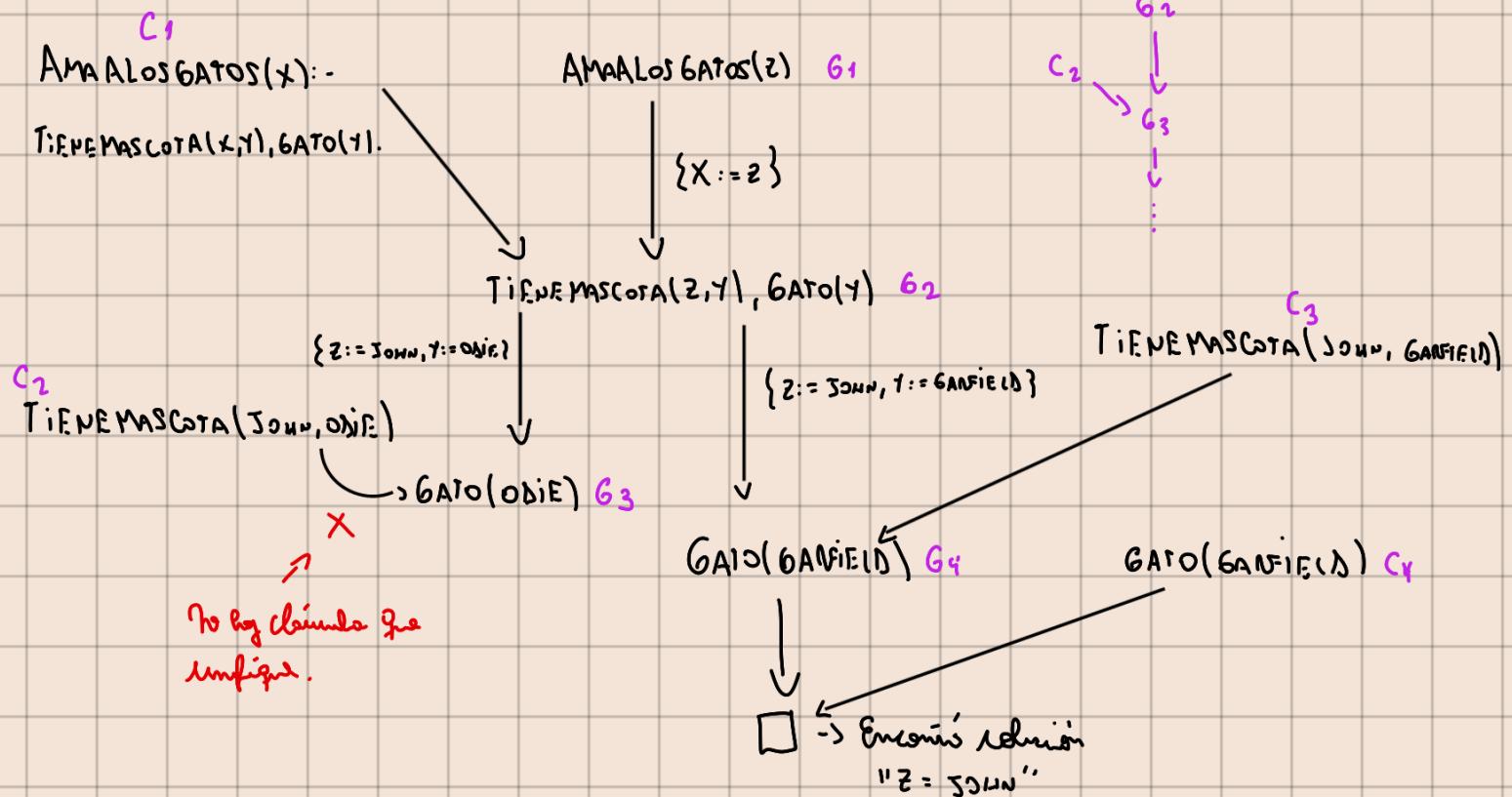
TIENE MASCOTA(John, ODIE).

TIENE MASCOTA(John, GANFIELD).

AMA ALOS GATOS(x) :- TIENE MASCOTA(x, y), GATO(y).

Cláusulas: D₁...D_m G₁

Realice el órden de



POSIBLES RESULTADOS DE CONSULTAS:

- TRUE: La reducción terminó en la cláusula nula.
- FALSE: La reducción terminó en una cláusula que no unificó con ninguna regla del programa.
- Algo más.

PAT. DE INSTANCIACIÓN: En un predicado no hay parámetros ni de entrada ni de salida.

+ x: Doblé elión instanciado

- x: Te doble elión instanciado

?X: Puede o no ser enviadas

MOTOR DE OPERACIONES ARITMÉTICAS: Es INDEPENDIENTE del MOTOR LÓGICO. Es EXTRALÓGICO.

Expresiones ARITMÉTICAS:

- Números
- Una variable de memoria
- $E_1 + E_2, E_1 - E_2$, siendo E_1, E_2 lfp. Aritméticas

Las OPERACIONES ARITMÉTICAS requieren que las EXP. estén INTEGRADAS.

OPERACIONES ARITMÉTICAS:

- $E_1 < E_2, E_1 = < E_2, E_1 > = E_2, E_1 =:: E_2, E_1 = \mid = E_2$: Encuentra ARGOS EXP

Aritmetizar y realiza la comparación.

- X is E: Puede E j verifica si X verifica con E.
 - ↳ 1 is 0+1 ✓
 - 1 is 1+1 ✗
 - 1+1 is 2 ✗

OPERACIONES NO ARITMÉTICOS:

- X = Y: En TNP si j reb x i X verifica con Y.
- X \= Y: En TNP si j reb x i X NO verifica con Y. ARGOS DEBEN ESTAR INSTANCIADOS.