ALGORITMO DE BÚSQUEDA A*

```
function BUSQUEDA-PRIMERO-EL-MEJOR (problema, eval-FN)
    FN-encolamiento ← función que ordena nodos por eval-FN
    return BUSQUEDA-GRAL (problema, eval-FN)

function BUSQUEDA-A* (problema)
    return BUSQUEDA-PRIMERO-EL-MEJOR(problema, g+h)

function BUSQUEDA-GRAL (problema, estrategia)   ← en árbol
```

⊛ Búsqueda en árbol NO ELIMINA ESTADOS REPETIDOS

```
Iterar:
    If (lista_abierta vacía):
        return fallo
    Else:
        elegir de lista_abierta, de acuerdo con estrategia, nodo a expandir
        If (nodo satisface test-objetivo):
            return solución
        Else:
            eliminar nodo de lista_abierta
            expandir nodo
            añadir nodos hijos a lista_abierta
```

⊛ A* sin eliminación de estados repetidos (búsqueda en árbol)

PROBLEMA

estado-inicial

lista-abierta (lista de estados descubiertos pero no explorados)

función → test-objetivo(estado) evalúa a verdadero o falso.

TIPOS DE ESTADOS:

I) estados descubiertos y explorados

II) descubierto + no explorado

III) no descubierto

función → estrategia (para insertar estados en lista_abierta de forma que esta mantenga un cierto orden

función → expandir

ESTADO ≡ nodo de búsqueda → información → { - nodo físico; - f, g, h ∈ $\mathbb{R}^+$; - ref al padre }

g: COSTE REAL ESTADO INICIAL → ESTADO actual

h: ESTIMACIÓN (HEURÍSTICA) DEL COSTE DEL CAMINO ÓPTIMO ESTADO actual → OBJETIVO

f: g + h → ESTIMACIÓN COSTE ÓPTIMO ESTADO INICIAL → META

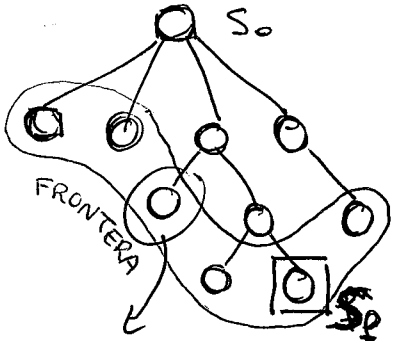
TRANSP. 12

```
function busqueda_en_arbol (problema, estrategia)
    Inicializar árbol_de_búsqueda con nodo_raíz
    Inicializar lista_abierta con nodo_raíz
    Iterar
        If (lista_abierta vacía):
            return [FALLO]
        Else:
            Elegir, de acuerdo con estrategia, un nodo a expandir de entre
            los que estan en lista_abierta
            If (nodo satisface test_objetivo)
                return [SOLUCIÓN]
            Else:
                Eliminar nodo de lista_abierta
                Expandir nodo
                Añadir nodos hijo a lista_abierta
```

⊛ Seguir el algoritmo TAL CUAL.

ESTRATEGIA = como ordeno nodos en lista_abierta

demostración Tma 1 (búsqueda en árbol)



Suponemos que A^* encuentra en este caso una solución no óptima.

SOLUCIÓN ÓPTIMA: $f^*(n) = g^*(n) + h^*(n)$

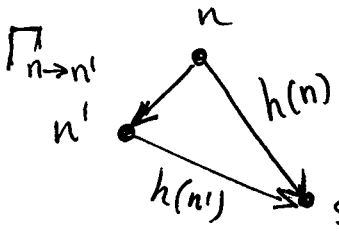
suponemos que este nodo en parte del camino óptimo

S_f NO ÓPTIMO $f_1 = g + h$ en S_f $h = 0$ siempre $f = g$

$\Rightarrow f = g(n_2) + h(n_2) \leq g^*(n_2) + h^*(n_2) \leq f_1$ contradicción

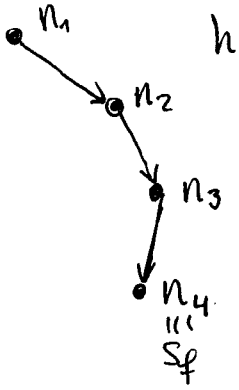
Para que A^* expanda este nodo de los ~~n~~ nodos hoja, este tiene que tener el menor f.

demostración Tma 3



$h(n) \leq \Gamma_{n\to n'} + h(n') \quad \forall n, n'$

(definición de monotonía)



$h(n_1) \leq \Gamma_{n_1\to n_2} + h(n_2)$

$h(n_2) \leq \Gamma_{n_2\to n_3} + h(n_3)$

$\Rightarrow h(n_1) \leq \Gamma_{n_1\to n_2} + \Gamma_{n_2\to n_3} + h(n_3)$

$h(n_3) \leq \Gamma_{n_3\to n_4} + h(n_4)$

$\Rightarrow h(n_1) \leq \Gamma_{n_1\to n_2} + \Gamma_{n_2\to n_3} + \Gamma_{n_3\to n_4} + \cancel{h(n_4)}$ (0)

$\Rightarrow h(n_1) \leq$ suma de costes en el camino que va de n a S_f (sea camino óptimo o no) $\Rightarrow h(n_1) \leq h^*(n_1)$

↑ entre esos caminos está el óptimo

POSIBLES ORDENACIONES (ESTRATEGIA)

→ de lista abierta

i) profundidad: 1º menos profundo } Búsqueda en anchura
ii) FIFO (cola)
iii) LIFO (pila) } Búsqueda en profundidad
iv) profundidad: 1º más profundo
v) por h: 1º menor h ← Búsqueda codiciosa
vi) por g: 1º menor g ← Búsqueda de coste uniforme
vii) por $f = g + h$: 1º menor f ← Búsqueda A^*

HEURÍSTICAS ADMISIBLES ÓPTIMAS: $\forall n \quad h(n) \leq h^*(n)$

$h^*(n) \equiv$ COSTE REAL DEL CAMINO ÓPTIMO ESTADO ACTUAL → META

TEOREMA 1: A^* (→ BÚSQUEDA EN ÁRBOL) SIN ELIMINACIÓN DE ESTADOS REPETIDOS CON HEURÍSTICA ADMISIBLE ES COMPLETA (encuentra la solución) Y ÓPTIMA (la solución es una de las de menor coste (podría haber más de una con el mismo coste óptimo)).

Observación: COSTES POSITIVOS (> 0) Y ADITIVOS.

TEOREMA 2: A^* (→ BÚSQUEDA EN GRAFO (lista abiertos y cerrados)) CON ELIMINACIÓN DE ESTADOS REPETIDOS CON HEURÍSTICA MONÓTONA ES COMPLETA Y ÓPTIMA.

HEURÍSTICA MONÓTONA: $\forall n, n'$ (n' es sucesor de n) $h(n) \leq \Gamma_{(n \to n')} + h(n'$

$\Gamma_{(n \to n')}$: coste de la acción que genera n' a partir de n.

TEOREMA 3: h MONÓTONA $\Rightarrow$ h ADMISIBLE

estado inicial

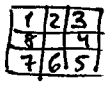
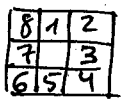
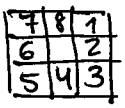
	1	5
3	2	6
4	7	8

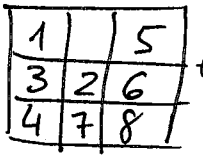
tipo 1 (de tipos de estados)

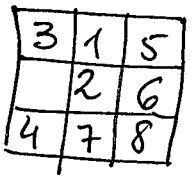
estado final (ejemplo)

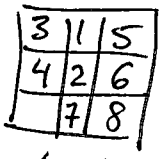
1	2	3
8		4
7	6	5

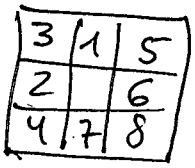
conjunto de soluciones:

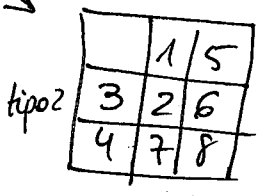
tipo 3 {  ,  ,  , ... }





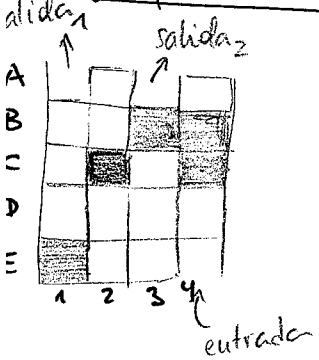






estado ≡ nodo físico del grafo
≡
nodo de búsqueda

Ejemplo laberinto



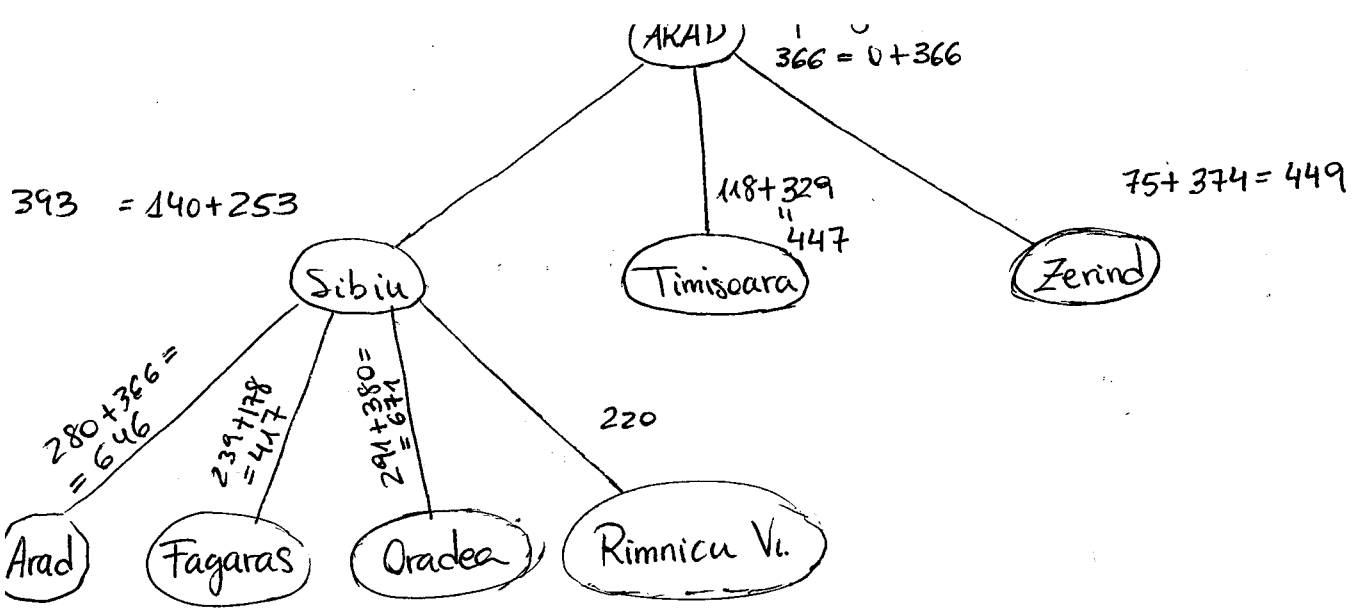
ACCIONES ↑ ← → ↓ moverse casilla adyacente

coste de acciones > 0
ADITIVOS

- g: COSTE REAL [estado inicial] ⇝ [estado actual]
- h: COSTE ESTIMADO DEL CAMINO ÓPTIMO [estado actual] ⇝ [estado final]
  - ↑ valor de la HEURÍSTICA

→ f = g + h
→ COSTE ESTIMADO [estado inicial] ⇝ [estado actual] ⇝ [estado final]

Vamos a usar h(n) ≡ distancia Manhattan (distancia L_1)
→ movimientos horizontales y verticales sin obstáculos.



$n \to n'$	$h(n)$	$\Gamma_{(n \to n')} + h(n')$
Neamt → Iasi	234	< 87 + 226
Iasi → Vaslui	226	< 92 + 199
Timisoara → Lugoj	329	< 111 + 244

ch admisible?

	n	$h(n)$	$h^*(n)$
E.I.	A	1	5
	B	1	3
	C	4	4
E.F.	D	0	0

1ETA

PODA α-β $[\alpha, \beta]$

en MAX $\alpha \longleftarrow \max(\alpha, \beta$'s sucesores$)$

en MIN $\beta \longleftarrow \min(\beta, \alpha$'s sucesores$)$

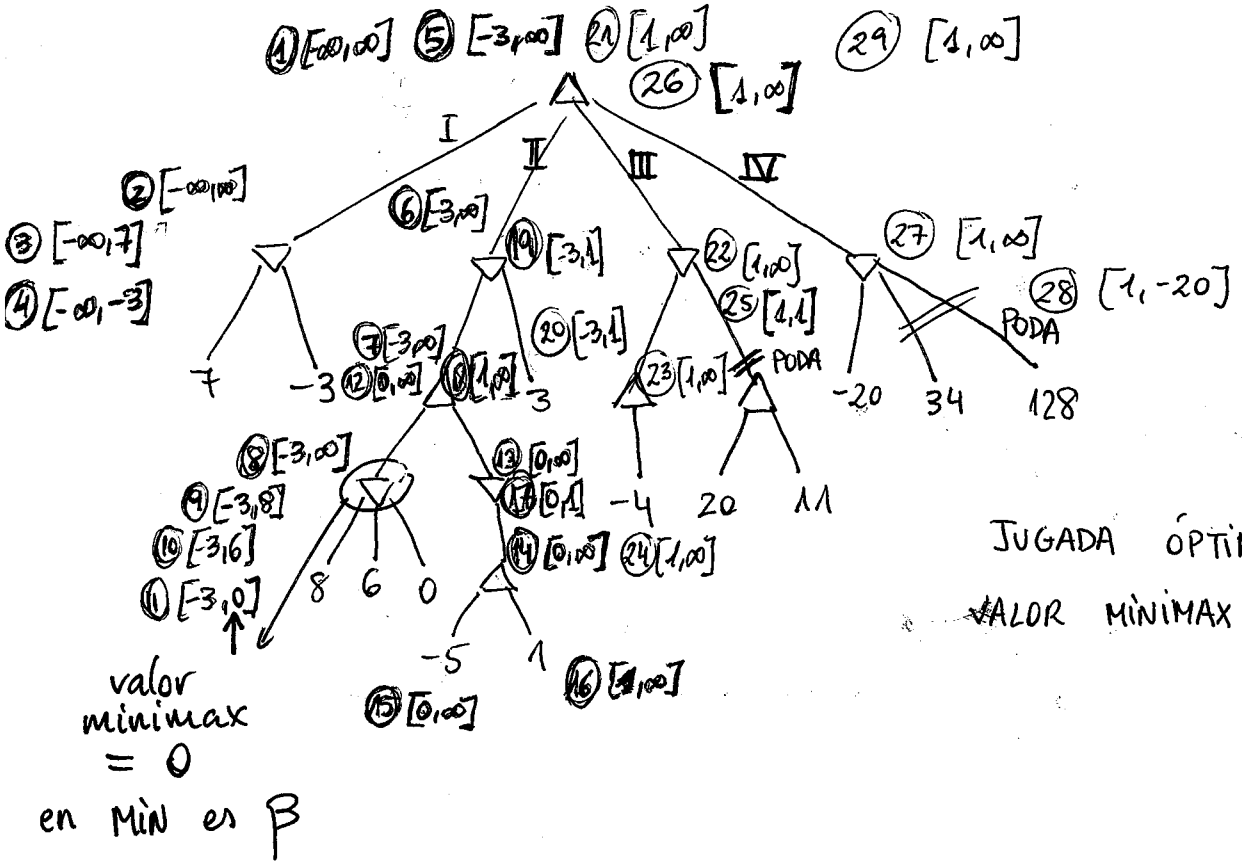
$\alpha \leq$ VALOR MINIMAX $\leq \beta$

PODA CUANDO $\alpha \geq \beta$

$[\alpha, \beta]$ intervalo de incertidumbre (contiene el valor minimax)

Valores minimax → en (MAX) es α
→ en (MIN) es β

MINIMAX CON PROMEDIO PROBABILISTA
EXCEPTI-MINIMAX



JUGADA ÓPTIMA II

VALOR MINIMAX EN RAÍZ: +1

EXAMEN:

- Formalización A^*
- Algoritmo poda α-β
- Formalización de predicados

$[\alpha, \beta]$

NODO MAX (Δ)

- cambiar α
- prop $\Uparrow$ α

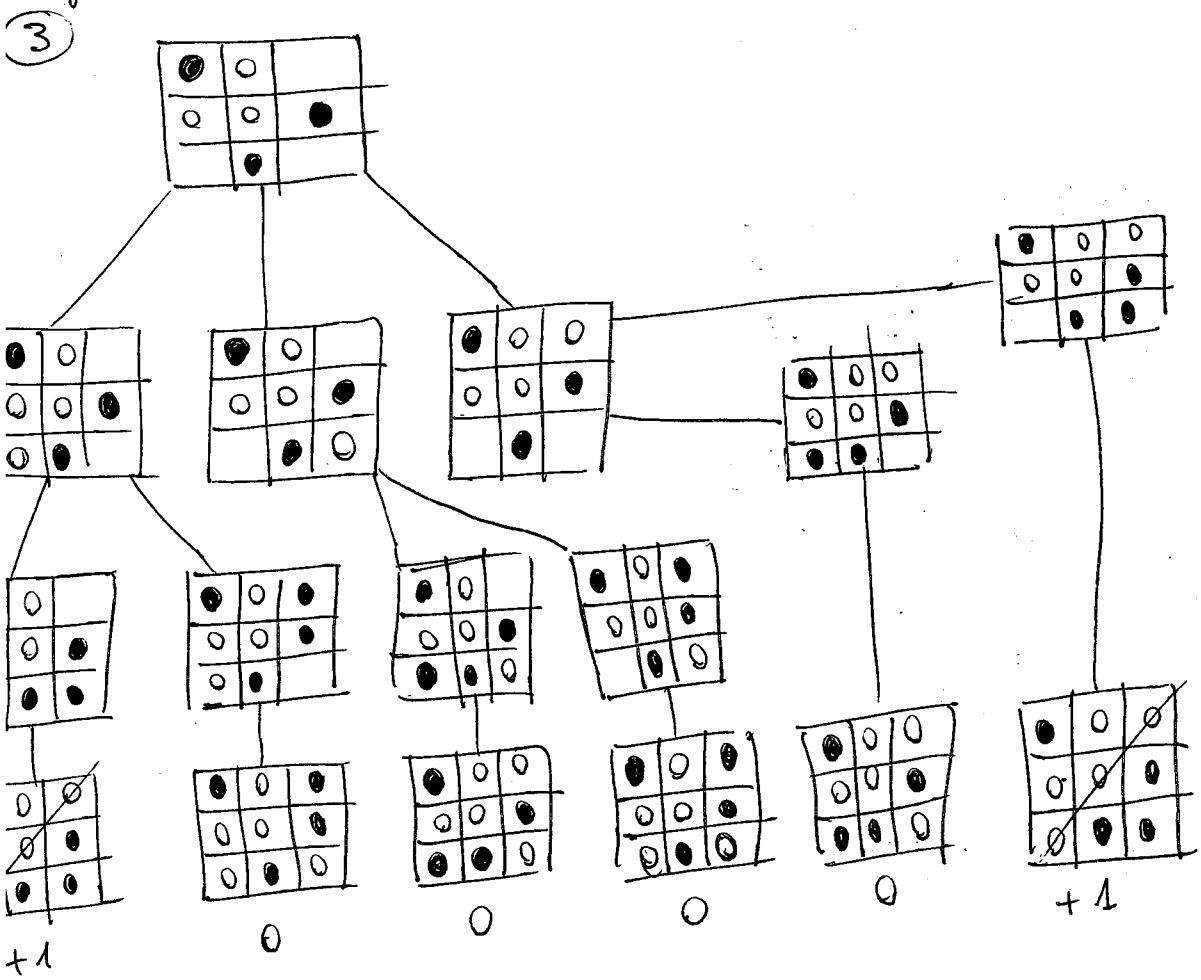
NODO MIN (∇)

- cambiar β
- prop $\Uparrow$ β

$\Downarrow [\alpha, \beta]$

PODAMOS si $\alpha \geq \beta$

Ejemplo: hoja 1 - parte final



Valor MINIMAX = 0

Rama = cualquiera de las tres (si consideramos MIN inteligente) elegida

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;    Lab assignment 2: Search
;;    LAB GROUP:
;;    Couple:
;;    Author 1:
;;    Author 2:
;;
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;    Problem definition
;;
(defstruct problem
  states              ; List of states
  initial-state       ; Initial state
  f-h                 ; reference to a function that evaluates to the
                      ; value of the heuristic of a state
  f-goal-test         ; reference to a function that determines whether
                      ; a state fulfils the goal
  f-search-state-equal ; reference to a predictate that determines
whether
                      ; two nodes are equal, in terms of their search
state
  operators)          ; list of operators (references to functions) to
                      ; generate successors
;;
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;    Node in search tree
;;
(defstruct node
  state          ; state label
  parent         ; parent node
  action         ; action that generated the current node from its
parent
  (depth 0)      ; depth in the search tree
  (g 0)          ; cost of the path from the initial state to this node
  (h 0)          ; value of the heurstic
  (f 0))         ; g + h
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;    Actions
;;
(defstruct action
  name              ; Name of the operator that generated the action
  origin            ; State on which the action is applied
  final             ; State that results from the application of the
action
  cost )            ; Cost of the action
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;    Search strategies
;;
(defstruct strategy
  name              ; name of the search strategy
  node-compare-p)   ; boolean comparison
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;    END: Define structures
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;    BEGIN: Define galaxy
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defparameter *planets* '(Avalon Davion Katril Kentares Mallory
Proserpina Sirtis))

(defparameter *white-holes*
  '((Avalon Mallory 6.4) (Avalon Proserpina 8.6)...))

(defparameter *worm-holes*
  '((Avalon Kentares 4) (Avalon Mallory 9)
    (Davion Katril 5) (Davion Sirtis 8)
    (Kentares Avalon 4) (Kentares Proserpina 12) ...))

(defparameter *sensors*
  '((Avalon 15) (Davion 5) ...))

(defparameter *planet-origin* 'Mallory)
(defparameter *planets-destination* '(Sirtis))
(defparameter *planets-forbidden*   '(Avalon))
(defparameter *planets-mandatory*   '(Katril Proserpina))
```

```
defun  insertar-prof (nodo-insertar lst)
    if:
        <= (cons  nodo-insertar lst)
    else:
        > (cons (first lst)
                (insertar-prof nodo-insertar (rest lst))))
```

Ama(x,y) ≡ x ama a y

Mata(x,y) ≡ x mata a y

Animal(x) ≡ x es un animal

Gato(x) ≡ x es un gato

~~Animal(x) ≡ x es animal~~ ~~x ≠ y~~

Tuno

Javier

Curiosidad

---

① $\forall x,z \left(\text{Animal}(x) \Rightarrow \left(\exists y \left(\text{Ama}(y,x) \right) \right) \right)$

② $\left(\text{Animal}(z) \wedge \text{Ama}(x,z) \right) \Rightarrow$

$\forall x,z \left[\left(\text{Animal}(z) \wedge \text{Mata}(x,y) \right) \Rightarrow \neg\exists y \left(\text{Ama}(y,x) \right) \right]$

③ $\forall x \left(\text{Animal}(x) \Rightarrow \text{Ama}(\text{Javier}, x) \right)$

④ ~~Mal~~ $\left[\text{Mata}(\text{Javier}, \text{Tuno}) \vee \text{Mata}(\text{Curiosidad}, \text{Tuno}) \right]$

---

① $\forall x,y \left[\left(\text{Animal}(y) \Rightarrow \text{Ama}(x,y) \right) \Rightarrow \left(\exists z \left(\text{Ama}(z,x) \right) \right) \right]$

② $\forall x \left[\left[\exists y \left(\text{Animal}(y) \wedge \text{Mata}(x,y) \right) \right) \Rightarrow \exists z \left(\neg\exists z \ \text{Ama}(z,x) \right) \right] \vee$

③ $\forall x \left[\text{Animal}(x) \Rightarrow \text{Ama}(\text{Javier}, x) \right] \vee$

④ ~~$\exists$~~ ~~$\text{Mata}(\text{Javier}, \text{Gato}(\text{Tuno})) \vee \text{Mata}$~~

~~$\forall \text{Tuno} \left[\text{Gato}(\text{Tuno}) \Rightarrow \text{Mata}(\text{Javier}, \text{Tuno}) \vee \text{Mata}(\text{Curiosidad}, \text{Tuno}) \right]$~~

$\text{Gato}(\text{Tuno}) \wedge \left[\text{Mata}(\text{Javier}, \text{Tuno}) \vee \text{Mata}(\text{Curiosidad}, \text{Tuno}) \right] \vee$

$\forall x \left[\forall y \left(Animal(y) \Rightarrow Ama(x,y) \right) \Rightarrow \exists z \; Ama(z,x) \right]$

$\forall x \left[\exists y \left(Animal(y) \wedge Mata(x,y) \right) \Rightarrow \neg \exists y \; Ama(z,x) \right]$

$\forall x \left[Animal(x) \Rightarrow Ama(Javier, x) \right]$

$Gato(Tuno)$ ← esto es un and

$Mata(Javier, Tuno) \vee Mata(Curiosidad, Tuno)$

$\forall x \; Gato(x) \Rightarrow Animal(x)$

$Mata(Curiosidad, Tuno)$???

Los amigos de mis amigos (A) son mis amigos (B) Si $A \Rightarrow B$

Si x es amigo mío e y amigo de x $\Rightarrow$ y es amigo mío

$\forall x, y \left[\left(AmigoDe(x,y) \wedge Amigo(y, Yo) \right) \Rightarrow Amigo(x, Yo) \right]$

Solo los pacifistas son ecologistas

$\forall x \left(EsEcologista(x) \Rightarrow EsPacifista(x) \right)$

Hay ecologistas ateos ≡ "Hay al menos 1 ecologista ateo"

~~$\forall x$~~ $\exists x \left(EsEcologista(x) \wedge EsAteo(x) \right)$

Hay al menos dos ecologistas ateos

$\exists x, y \left(EsEcologista(x) \wedge ESEcologista(y) \wedge Ateo(x) \wedge Ateo(y) \wedge \neg Igual(x,$

Los locales solo hablan mal de los forasteros que no conocen

$\forall x, y \left[\left(F(y) \wedge L(x) \wedge H(x,y) \right) \Rightarrow \neg((x,y) \right]$