Inteligencia Artificial 2017-2018

Práctica 2: LISP++

Grupo: 2213

Celia San Gregorio Moreno

Álvaro Martínez Morales

Índice

[MODELIZACIÓN DEL PROBLEMA 3](#_Toc509768256)

[Ejercicio 1: Evaluación del valor de la heurística 3](#_Toc509768257)

[Código 3](#_Toc509768258)

[Comentario 3](#_Toc509768259)

[Ejercicio 2: Operadores navigate-worm-hole y navigate-white-hole 4](#_Toc509768260)

[Código 4](#_Toc509768261)

[Comentario 5](#_Toc509768262)

[Ejercicio 3A: Test para determinar si se ha alcanzado el objetivo 6](#_Toc509768263)

[Código 6](#_Toc509768264)

[Comentario 7](#_Toc509768265)

[Ejercicio 3B: Predicado para determinar la igualdad entre estados de búsqueda 8](#_Toc509768266)

[Código 8](#_Toc509768267)

[Comentario 8](#_Toc509768268)

[FORMALIZACIÓN DEL PROBLEMA 9](#_Toc509768269)

[Ejercicio 4: Representación LISP del problema 9](#_Toc509768270)

[Código 9](#_Toc509768271)

[Comentario 9](#_Toc509768272)

[Ejercicio 5: Expandir nodo 10](#_Toc509768273)

[Código 10](#_Toc509768274)

[Comentario 10](#_Toc509768275)

[Ejercicio 6: Gestión de nodos 11](#_Toc509768276)

[Código 11](#_Toc509768277)

[Comentario 11](#_Toc509768278)

[Ejercicio 7: Definir estrategia para la búsqueda A\* 12](#_Toc509768279)

[Código 12](#_Toc509768280)

[Comentario 12](#_Toc509768281)

[Ejercicio 8: Función de búsqueda 13](#_Toc509768282)

[Código 13](#_Toc509768283)

[Comentario 13](#_Toc509768284)

[Ejercicio 9: Ver el camino seguido y la secuencia de acciones 14](#_Toc509768285)

[Código 14](#_Toc509768286)

[Comentario 14](#_Toc509768287)

[Ejercicio 10: Otras estrategias de búsqueda 15](#_Toc509768288)

[Código 15](#_Toc509768289)

[Comentario 15](#_Toc509768290)

[Ejercicio 11: Ejercicios de reflexión 16](#_Toc509768291)

# MODELIZACIÓN DEL PROBLEMA

# Ejercicio 1: Evaluación del valor de la heurística

### Código

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

;; BEGIN: Exercise 1 -- Evaluation of the heuristic

;;

;; Returns the value of the heuristics for a given state

;;

;; Input:

;; state: the current state (vis. the planet we are on)

;; sensors: a sensor list, that is a list of pairs

;; (state cost)

;; where the first element is the name of a state and the second

;; a number estimating the cost to reach the goal

;;

;; Returns:

;; The cost (a number) or NIL if the state is not in the sensor list

;;

(defun f-h-galaxy (state sensors)

(second (assoc state sensors)))

;;

;; END: Exercise 1 -- Evaluation of the heuristic

;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

### Comentario

El ejercicio nos pedía calcular el valor de la heurística en un estado dado.

Dado que este cálculo ya viene *‘hardcodeado’* en la estructura de la implementación base de la práctica, la única codificación necesaria fue la extracción de ese dato de la estructura de los sensores.

# Ejercicio 2: Operadores *navigate-worm-hole* y *navigate-white-hole*

### Código

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

;; BEGIN: Exercise 2 -- Navigation operators

;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

;; Crea una lista de tripletes con state como planeta de origen

;; y cada uno de los planetas en el agujero blanco o de gusano

;; a los que se puede acceder desde state.

;;

;; Input:

;; state: el estado actual (el planeta donde estamos)

;; hole-map: lista de tripletes correspondiente a los grafos

;; de la galaxia (en este caso, agujeros blancos

;; o de gusano).

;;

;; Returns:

;; Lista de tripletes de tipo (<state> <planeta-destino> <coste>).

(defun make-colindant-list (state hole-map)

;Si hemos llegado al final de la lista

;asociativa, la función termina.

(if (null hole-map)

nil

;Si no, comprueba si el planeta de origen (state)

;coincide con el planeta de origen del primer triplete.

(if (equal state (first (first hole-map)))

;Si coincide, crea una lista de tripletes.

(cons (first hole-map)

(make-colindant-list state (rest hole-map)))

;Sino, avanza en la lista asociativa hole-map.

(make-colindant-list state (rest hole-map)))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

;; Operador genérico que devuelve una lista de acciones que se

;; pueden hacer a partir del estado state, sobre un

;; grafo cualquiera, con posibildad de exclusión.

;;

;; Input:

;; state: estado de búsqueda que representa al planeta de origen.

;; hole-map: lista de tripletes correspondiente al grafo cualquiera.

;; forbidden: planetas que no permitir como destino. De no haberlos, debe ser nil

;; action-name: nombre que asignar a la acción.

;;

;; Returns:

;; Lista de acciones de la acción definida del planeta de origen al de destino.

(defun navigate (state hole-map forbidden action-name)

;Genera una lista de acciones con los resultados de

;la función 'make-colindant-list' no presentes en forbidden.

(mapcan #'(lambda (dest)

(if (member (second dest) forbidden)

nil

(list (make-action

:name action-name

:origin state

:final (second dest)

:cost (third dest)))))

(make-colindant-list state hole-map)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

;; Operador que devuelve una lista de acciones que se

;; pueden hacer a partir del estado state, sobre un

;; grafo con agujeros blancos.

;;

;; Input:

;; state: estado de búsqueda que representa al planeta de origen.

;; white-holes: lista de tripletes correspondiente al grafo de

;; agujeros blancos de la galaxia.

;;

;; Returns:

;; Lista de acciones del planeta de origen al de destino, a

;; través de los agujeros blancos.

(defun navigate-white-hole (state white-holes)

(navigate state white-holes nil 'navigate-white-hole))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

;; Operador que devuelve una lista de acciones que se

;; pueden hacer a partir del estado state, sobre un

;; grafo con agujeros de gusano.

;;

;; Input:

;; state: estado de búsqueda que representa al planeta de origen.

;; white-holes: lista de tripletes correspondiente al grafo de

;; agujeros de gusano de la galaxia.

;;

;; Returns:

;; Lista de acciones del planeta de origen al de destino, a

;; través de los agujeros de gusano.

(defun navigate-worm-hole (state worm-holes planets-forbidden)

(navigate state worm-holes planets-forbidden 'navigate-worm-hole))

;;

;; END: Exercise 2 -- Navigation operators

;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

### Comentario

El ejercicio consistía en, dado un estado, elaborar un listado con las acciones que pudieran tener lugar desde el mismo en base a los distintos mapas estelares a nuestra disposición (en este caso, el mapa de *worm holes* y *white holes*. Esto, claro, implica la exclusión de los *‘planetas prohibidos’* en el caso de los *worm holes* como posibles destinos.

Esto se ha implementado, fundamentalmente, mediante dos funciones principales y una interfaz para cada tipo de mapa.

La *primera función principal* es **make-colindant-list**, que devuelve todos los estados colindantes a un estado dado en base al mapa indicado.

La *segunda función principal* es **navigate**, que, dado un estado, un mapa, una posible declaración de nodos prohibidos y el nombre de una acción, produce, mediante la primera función, una lista instancias de la acción indicada con cada nodo colindante en el que se excluyen los planetas prohibidos, de haberlos.

Por último, las interfaces son una capa de transparencia sobre la función principal *navigate*, que permite, aparte de especificar la acción para cada tipo de mapa, sólo pedir la información requerida para cada uno de los dos tipos de mapas.

# Ejercicio 3A: Test para determinar si se ha alcanzado el objetivo

### Código

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

;; BEGIN: Exercise 3A -- Goal test

;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

;; Comprueba si el nodo pasado como argumento es un estado objetivo.

;;

;; Input:

;; nodo: nodo que representa un estado de búsqueda (el planeta actual).

;; planets-destination: lista de nombres de los planetas destino.

;; planets-mandatory: lista de nombres de los planetas obligatorios.

;;

;; Returns:

;; T si el nodo es un estado objetivo, NIL si no.

(defun f-goal-test-galaxy (node planets-destination planets-mandatory)

;Si el nodo está entre la lista de planetas destino,

;comprueba que los nodos antecesores hayan pasado por

;los planetas obligatorios.

(if (member (node-state node) planets-destination)

;(f-mandatory-test node planets-mandatory)

(f-mandatory-test (node-parent node) planets-mandatory) ;Comprueba si los nodos padre corresponden

nil)) ;a planetas obligatorios visitados.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

;; Devuelve una lista de planetas obligatorios aún no visitados.

;;

;; Input:

;; node: nodo que representa un estado de búsqueda (el planeta actual).

;; planets-mandatory: lista de nombres de los planetas obligatorios.

;;

;; Returns:

;; Lista con los nombres de los planetas obligatorios que aún

;; queden por visitar, o NIL si se han visitado todos.

(defun get-mandatory-planets-not-visited (node planets-mandatory)

;Si llegamos al nodo raíz, devolvemos

;la lista de planetas que quedan por visitar.

(if (null node)

planets-mandatory

;Si aún no hemos llegado al nodo raíz, comprueba si el nodo actual

;es un planeta obligatorio.

(if (member (node-state node) planets-mandatory :test #'equal)

;Si es un planeta obligatorio, lo elimina de la lista y pasa a comprobar el nodo padre.

(get-mandatory-planets-not-visited (node-parent node) (remove (node-state node) planets-mandatory))

;Si no es obligatorio, pasa a comprobar el nodo padre directamente.

(get-mandatory-planets-not-visited (node-parent node) planets-mandatory))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;

;; Comprueba si en el camino del nodo raíz al nodo actual

;; se ha pasado por los planetas obligatorios.

;;

;; Input:

;; nodo: nodo que representa un estado de búsqueda (el planeta actual).

;; planets-mandatory: lista de nombres de los planetas obligatorios.

;;

;; Returns:

;; T si se ha pasado por todos los nodos obligatorios, NIL si no.

(defun f-mandatory-test (node planets-mandatory)

;Y la lista de planetas obligatorios está vacía,

;hemos pasado por todos los planetas obligatorios.

(if (null (get-mandatory-planets-not-visited node planets-mandatory))

T

nil))

­

;;

;; END: Exercise 3A -- Goal test

;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

### Comentario

El ejercicio

# Ejercicio 3B: Predicado para determinar la igualdad entre estados de búsqueda

### Código

;;

### Comentario

El ejercicio

# FORMALIZACIÓN DEL PROBLEMA

# Ejercicio 4: Representación LISP del problema

### Código

;;

### Comentario

El ejercicio

# Ejercicio 5: Expandir nodo

### Código

;;

### Comentario

El ejercicio

# Ejercicio 6: Gestión de nodos

### Código

;;

### Comentario

El ejercicio

# Ejercicio 7: Definir estrategia para la búsqueda A\*

### Código

;;

### Comentario

El ejercicio

# Ejercicio 8: Función de búsqueda

### Código

;;

### Comentario

El ejercicio

# Ejercicio 9: Ver el camino seguido y la secuencia de acciones

### Código

;;

### Comentario

El ejercicio

# Ejercicio 10: Otras estrategias de búsqueda

### Código

;;

### Comentario

El ejercicio

# Ejercicio 11: Ejercicios de reflexión

**1. ¿Por qué se ha realizado este diseño para resolver el problema de búsqueda?**

**1.1 ¿Qué ventajas aporta?   
1.2 ¿Por qué se han utilizado funciones lambda para especificar el test objetivo, la heurística y los operadores del problema?**

**2 Sabiendo que en cada nodo de búsqueda hay un campo “parent”, que proporciona una referencia al nodo a partir del cual se ha generado el actual ¿es eficiente el uso de memoria?**

**3 ¿Cuál es la complejidad espacial del algoritmo implementado?**

**4 ¿Cuál es la complejidad temporal del algoritmo?**

**5 Indicad qué partes del código se modificarían para limitar el número de veces que se puede utilizar la acción “navegar por agujeros de gusano” (bidireccionales).**