Inteligencia Artificial 2017-2018

Práctica 2: LISP++

Grupo: 2213

Celia San Gregorio Moreno

Álvaro Martínez Morales

Índice

[Ejercicio 1: Evaluación del valor de la heurística 2](#_Toc509764315)

[Código 2](#_Toc509764316)

[Comentario 2](#_Toc509764317)

[Ejercicio 2: Operadores navigate-worm-hole y navigate-white-hole 3](#_Toc509764318)

[Código 3](#_Toc509764319)

[Comentario 4](#_Toc509764320)

[Ejercicio 3: Test para determinar si se ha alcanzado el objetivo 5](#_Toc509764321)

[Código 5](#_Toc509764322)

[Comentario 5](#_Toc509764323)

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

### Código

;

### Comentario

El ejercicio