Examen2.pdf

Anónimo

Inteligencia Artificial

3º Grado en Ingeniería Informática

Escuela Superior de Ingeniería
Universidad de Cádiz

Accede al documento original

NOMBRE Y APELLIDOS: _____

Ejercicios en Blanco: 1 2 3 4 5 6 (Rodea con un círculo los ejercicios no entregados)

TIEMPO DE REALIZACIÓN: 3 HORAS

Se ha de entregar **cada ejercicio en folios distintos**. Cada folio debe incluir el **nombre del alumno y el número** del ejercicio **y estar numerado cuando hay más de una hoja por ejercicio**.

1. Formalización Problema de Juegos entre adversarios (3 puntos)

Considérese el siguiente juego entre 2 adversarios que utiliza el tablero que se muestra a continuación. Donde cada jugador pone una marca cuando ocupa una casilla, X para Max, O para Min.

1	2	3	4	5	6	7	8	9

El objetivo es ocupar tres casillas que sumen 15. Para ello en cada turno un jugador ocupa una de las celdas numeradas, y así bloquea esa celda al otro jugador, por ejemplo, si Max ocupa la celda 9, Min ya no puede usar esa celda. Se pueden seleccionar más de 3 celdas. Por ejemplo, si Max ha seleccionado las celdas 1, 2, 5 y 9 gana, pues de las 4 que ha ocupado con las celdas 1, 5 y 9 suma 15.
El juego finaliza cuando:

- Gana algún jugador que haya ocupado tres números distintos que sumen 15. No hace falta que el tablero esté completo.
- Se llene el tablero pero ninguno haya ganado.

A. Realiza la formalización como un problema cuya resolución puede realizarse mediante búsqueda entre adversarios (en C) **con el formato de funciones estudiado en la asignatura, incluyendo al menos:**
   - Constantes y tipo de datos apropiado para los estados y utilízalo para definir el estado inicial para este problema.
   - Función esVálida que determina si es o no posible la aplicación de cada jugada a partir de un estado concreto.
   - Función aplicaJugada que lleva a cabo la aplicación de cualquiera de las posibles jugadas para MAX.
   - Función Terminal para comprobar si se ha llegado a un estado terminal.
   - Función Utilidad que devuelva los valores de utilidad correspondientes a que gane MAX o MIN o queden empatados.

2. Aplicación Poda alfa-beta (1 punto)

Construye el árbol de búsqueda completo suponiendo el siguiente estado inicial, y siendo el turno de MAX.

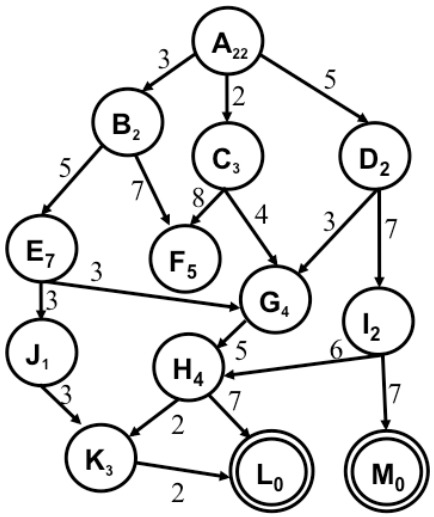
MIN	MAX							
1	2	3	4	5	6	7	8	9

Indica cuál es la jugada más prometedora que se obtiene tras aplicar sobre el árbol anterior, la estrategia **Minimax con poda alfa-beta**. (describe paso a paso cómo se van asignando los valores a los nodos y cómo se va realizando la poda)

3. Búsqueda HEURÍSTICA (1 punto)

El siguiente grafo representa un problema de espacio de estados, donde los nodos representan los estados del problema, los arcos el coste real de ir de un nodo a otro, y el número junto al nombre de cada nodo representa el valor de la heurística en cada estado. El estado inicial es A y los estados finales son L y M.

A. Determina qué solución y qué coste real se obtienen aplicando **el algoritmo de búsqueda A*** Especifica el proceso de búsqueda paso a paso, estableciendo en cada iteración cuál es el nodo Actual, y el contenido de las listas de nodos Abiertos y Cerrados junto con los valores de la función de evaluación. En caso de empate, los nodos se almacenan y extraen de cualquier lista en Orden Alfabético



4. RETE (1,5 puntos)

Dado la siguiente Base de Reglas y la Base de Hechos:

- Construye la Red de Redundancia Temporal (RETE).
- Realiza una simulación de la ejecución correspondiente con la red construida y aplicando las siguientes estrategias de Resolución de Conflictos:
  a. Especificidad
  b. Refracción

*Al igual que la acción Assert se representa con el símbolo (+), la acción Modify se representa con los símbolos (**+-**).*

Base de Reglas

```
(defrule R1
 ?f1<-(valvula (nombre ?v) (tempe1 ?t1)(tempe2 ?t2)
               (presion ?p1) (estado abierta))
 ?f2<-(alarma (nombre ?v) (presion ?p2)
              (limite_tempe ?lt) (estado ON))
 (test (>=  ?p1  ?p2))
=> (modify ?f1 (presion 0) )
)
```

```
(defrule R2
  ?f1<- (valvula (nombre ?v) (tempe1 ?t1) (tempe2 ?t2)
                 (presion ?p) (estado cerrada))
  (test (<= ?p 5))
  (test (> ?t2 ?t1))
=>
   (modify ?f1 (estado abierta) )
   (assert (alarma (nombre ?v) (presion ?p)
                   (limite_tempe ?t1) (estado ON)))
)
```

```
(defrule R3
    ?f1<- (valvula (nombre ?v1)  (tempe1 ?t1) (tempe2 ?t12) (presion 0) (estado abierta))
    ?f2<- (valvula (nombre ?v2) (tempe1 ?t1) (tempe2 ?t22) (presion 0) (estado abierta))
    ?f3<- (alarma (nombre ?v1) (presion ?p3) (limite_tempe ?lt1) (estado ON))
    ?f4<- (alarma (nombre ?v2) (presion ?p4) (limite_tempe ?lt2) (estado ON))
    (test (neq ?v1 ?v2))
=>
    (modify ?f1 (presion (+ ?p3 ?p4)) (estado cerrada))
    (modify ?f2 (presion (+ ?p3 ?p4)) (estado cerrada))
    (retract ?f3 ?f4)
)
```

Base de Hechos

(valvula (nombre Salida) (tempe1 10) (tempe2 55) (presion 5) (estado cerrada)))
(valvula (nombre Pasillo) (tempe1 41) (tempe2 37) (presion 5) (estado abierta))
(valvula (nombre Entrada) (tempe1 10) (tempe2 35) (presion 2) (estado cerrada))

5. Implementación en CLIPS de un Sistema Basado en Reglas (2,5 puntos)

Se pretende implementar mediante un sistema basado en reglas un agente inteligente que realice de forma automática la compra en un supermercado. Para ello se dispone de un agente inteligente que se desplaza por los pasillos del supermercado con su propio carro buscando los productos de la lista de la compra. Supongamos los siguientes hechos que deberán definirse de forma estructurada mediante plantillas:

- Productos: contiene la información de los productos del supermercado: un identificador y un nombre, para guardar su ubicación se tendrá en cuenta el número del pasillo (numerados de 1 a 12), además se almacenará la cantidad de ese producto que hay en stock y el precio unitario.
- Pedidos: los clientes pueden realizar pedidos que en el sistema entrarán como hechos con la siguiente información: identificador del cliente que realiza el pedido, el producto y las unidades que desea comprar. Por ejemplo:

  (pedido (id_cliente 33) (id_producto 1) (unidades 3))

- Carro de la compra: existe un carro de la compra por cada cliente, contendrá el identificador del cliente, el número de artículos que lleva comprados, el importe total en euros así como el pasillo actual donde se encuentra el agente.

Las acciones que se van a realizar son las siguientes:

- **Asignar un carro**: cuando un cliente nuevo se registra en el sistema introduce su identificador (se realizará con instrucciones del tipo: (assert (nuevo_cliente id_cliente))
  y automáticamente se le asigna un carro que saldrá del pasillo 1, y la factura y número de productos comprados estará a 0.

A partir de este momento, los pedidos de este cliente se realizarán con instrucciones del tipo: (assert (pedido (id_cliente xx) (id_producto yy) …etc)

Las acciones que se podrán realizar entonces serán:

- **Mover carro:** se mueve el carro a un pasillo contiguo en orden ascendente, es decir, si el pasillo actual del carro es el 3, esta regla mueve el carro al pasillo 4. Cuando se encuentre en el pasillo 12, entonces pasará al pasillo 1.
- **Comprar:** se puede comprar un producto pedido cuando el carro se encuentre en el mismo pasillo que este producto y exista stock suficiente. El pedido realizado se borra, y se calcula el precio que se ha de añadir a la factura del carro. La cantidad en stock en el producto también debe actualizarse. La compra deberá tener prioridad sobre el desplazamiento del carro por el supermercado.
- **Existencias Insuficientes:** Cuando no hay stock suficiente para realizar la compra, se avisa al cliente con un mensaje en pantalla y se elimina este pedido.

A. Define las reglas necesarias para que el agente inteligente pueda llevar a cabo su labor de forma correcta .
B. Usa plantillas para definir las propiedades de las cajas y de los artículos.
C. Define y usa al menos una función para la realización de los cálculos aritméticos.
D. Realiza una traza de ejecución con los siguientes hechos de ejemplo:

```
(deffacts productos
 (producto   (id_producto 1) (nombre leche)(pasillo 3) (stock 45) (precio 1))
 (producto (id_producto 2) (nombre galletas) (pasillo 4)(stock  10) (precio 2.20))
 (producto (id_producto 3) (nombre cafe) (pasillo 5) (stock 2) (precio 2.6))
 (producto (id_producto 4) (nombre arroz) (pasillo 12) (stock 30) (precio 1.98)))

(assert (nuevo_cliente 33))
(assert (pedido (id_cliente 33) (id_producto 1) (unidades 3)))
(assert (pedido (id_cliente 33) (id_producto 2) (unidades 30)))
```

6. **Implementa en lenguaje C la estrategia de búsqueda informada A* de forma que pueda ser ejecutada con cualquiera de las formalizaciones de problemas de búsqueda vista en clase, en particular, con el 8-puzle.**

(código principal de la búsqueda A* y la función Expandir Nodos, junto con las funciones auxiliares necesarias para su correcta ejecución)