Inteligencia Artificial

Grado en Ingeniería Informática
Departamento de Ingeniería Informática

UCA

NOMBRE Y APELLIDOS: [redacted]

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

TIEMPO DE REALIZACIÓN: 3 HORAS

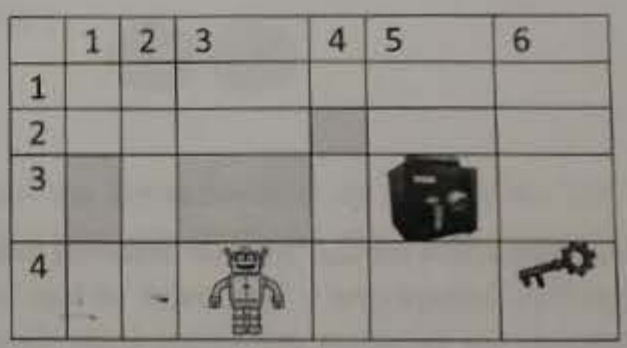
Se ha de entregar **cada ejercicio en folios distintos**. Cada folio debe incluir el **nombre del alumno y el número** del ejercicio **y estar numerado cuando hay más de una hoja por ejercicio**.

1. Formalización Problema de Búsqueda (3 puntos)

Un robot debe coger un objeto que se encuentra bajo llave en una caja blindada situada en alguna de las celdas de un tablero de N filas y M columnas. La llave no puede salir de la habitación donde se encuentra, por lo que debe arrastrar la caja hasta la llave. Hay celdas que no pueden ser visitadas (en la figura de ejemplo las celdas en gris). El robot se desplaza siempre a celdas adyacentes, debe primero dar con la caja y una vez tenga la caja podrá arrastrarla consigo hasta la llave y abrir la caja después. La búsqueda finaliza cuando todos los elementos se encuentran en la misma celda y la caja está abierta, lo cual indica que el robot va a coger el objeto.

Por tanto las acciones posibles que puede realizar el robot son las siguientes:

- Moverse él solo de una celda a otra contigua (izquierda, derecha, arriba o abajo) cuando no ha encontrado aún la caja.
- Arrastrar la caja de una celda a otra contigua (entonces la caja cerrada y el robot se desplazarán a la posición adyacente que corresponda) buscando la llave.
- Abrir la caja blindada del objeto (sólo si el robot, la caja cerrada y la lave están en la misma celda).



El problema debe plantearse como un problema de búsqueda en espacio de estados para encontrar el camino más corto dado cualquier estado inicial posible y que pueda por tanto, ser resuelto usando la implementación de cualquier estrategia de búsqueda de las vistas en clase, simplemente incorporando la formalización correspondiente (en C o en seudocódigo):

A. Describe el tipo de datos tEstado apropiado para este problema y utilízalo para definir el estado inicial y final para este ejemplo, a través de las funciones correspondientes.
B. Describe la lista de operadores que se contemplan para este problema.
C. Diseña la función esVálido que determina si es o no posible la aplicación de cada operador a partir de un estado concreto.
D. Diseña la función aplicaOperador que lleva a cabo la aplicación de cualquiera de los posibles operadores devolviendo un nuevo estado.
E. Diseña la función TestObjetivo para comprobar si se ha alcanzado el objetivo del problema de acuerdo al enunciado propuesto.

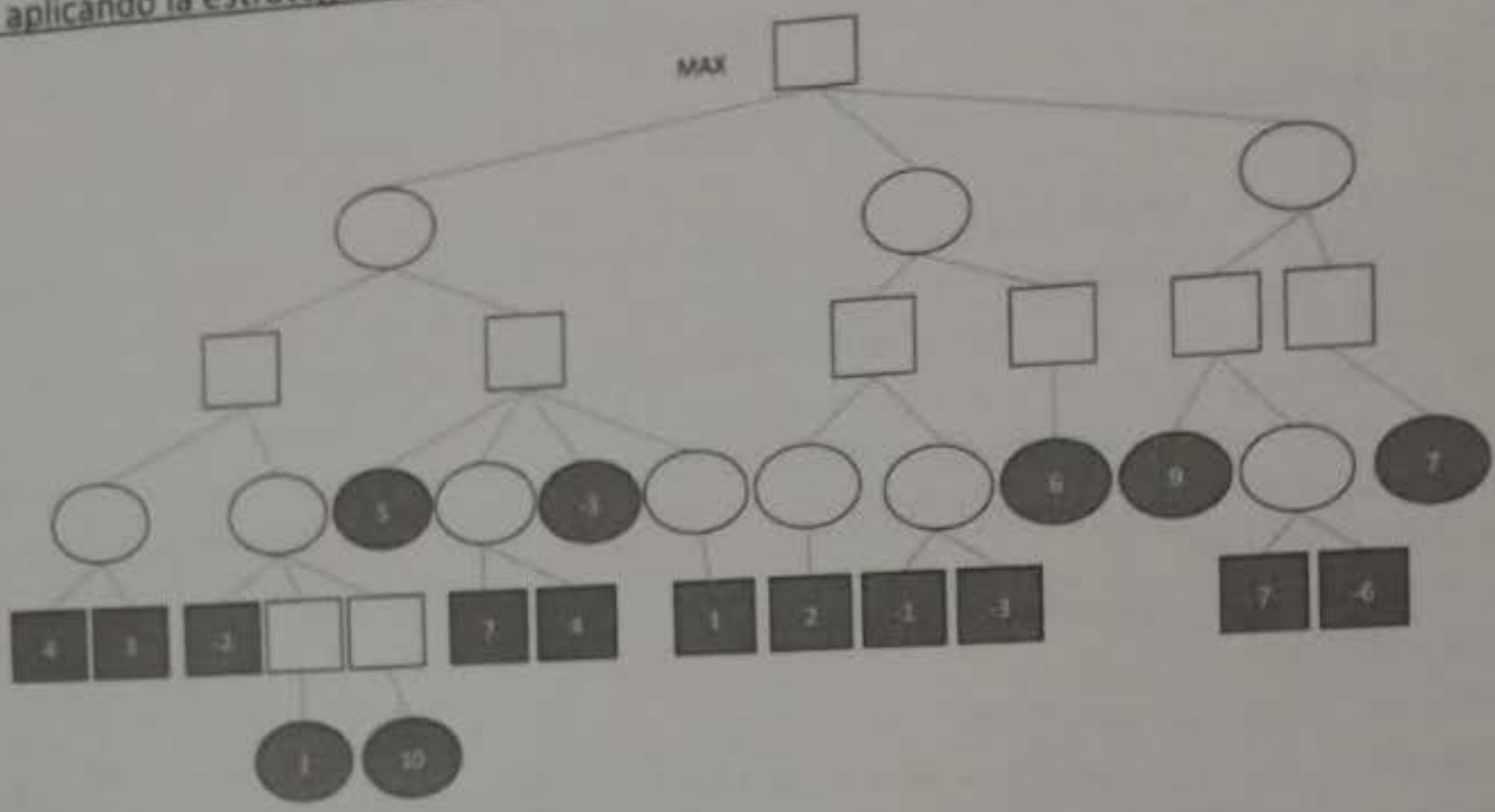
2. Aplicación de una estrategia de búsqueda (1,25 puntos)

Suponiendo el estado inicial dado en la figura 1,

- **Diseña una función heurística** y aplica la estrategia A* indicando en cada paso el nodo actual, el contenido de las listas (Abiertos, etc.) así como el árbol de búsqueda que se va generando.

3. Poda Alfa-beta (0,75 puntos)

Aplica la estrategia de **Poda alfa-beta** de Izquierda a Derecha al siguiente árbol genérico, donde los valores de la función de evaluación aparecen en cada nodo terminal del árbol.
Especifica los valores alfa y beta para cada nodo, cómo y **en qué orden** se van actualizando a medida que se va aplicando la estrategia de poda.



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

Las compañías Robaphone y Timoestar se han fusionado y han lanzado un nuevo sistema basado en reglas para gestionar los servicios prestados a los clientes. La información de los planes ofertados, los clientes, sus peticiones y su consumo se almacenarán en forma de hechos estructurados, mientras que a través de reglas se gestionarán ciertos servicios.

4.1. Define la plantilla y los hechos iniciales de los 3 planes de la empresa:

PLANES

La empresa cuenta con 3 planes para los clientes particulares que son los siguientes:

- **Plan Cotillea**: es el plan básico fijo+Internet 20Mb, no ofrece ni servicio de móvil ni tv. Sólo cobra los minutos de una llamada que exceda la hora de duración, a un precio de 2 céntimos por minuto. El precio total de la plan es de 60 Euros.
- **Plan Naufraga**: fijo + Internet 30 Mb + Móvil, datos móviles a 4GB. Cobra los minutos de las llamadas que excedan de la hora a 0 euros en el fijo, pero los del móvil a 5 céntimos/minuto. El precio total de la plan es de 80 Euros. Coste SMS a 2 céntimos.
- **Plan Ruina:** Igual que el Naufraga con Internet a 50Mb y ofrece 600 canales. Cobra todos los minutos demás tanto en fijos como en móviles a 3 céntimos/minuto y los SMS gratis. El precio total de la plan es de 95 Euros.

Para almacenar la información de estos planes crea una plantilla que se denomine Plan, y que contenga la siguiente información: nombre del plan (Cotillea, Naufraga o Ruina), la velocidad de la fibra, tarifa fijo y tarifa móvil para calcular el precio por minuto de las llamadas que excedan la hora, el coste sms precio de los sms, la velocidad de la red de datos móviles, el número de canales de tv disponibles. Por último el precio de la plan?.

4.2. Define las plantillas para los Clientes y las peticiones (Consumo viene dada)

CLIENTES
Crea la plantilla correspondiente para almacenar información sobre el id del cliente registrado en el sistema y el plan asignado, por defecto será el plan básico. Además contendrá un campo con la tarifa/minuto del Roaming, por defecto a 50 céntimos/minuto.

PETICIÓN
Crea la plantilla correspondiente a las peticiones que el cliente puede realizar, actualmente sólo sería un Cambio de Plan, debiendo indicar el id del cliente que realiza la petición, velocidad de la fibra, y si desea TV.

CONSUMO
Internamente el sistema va almacenando el consumo de cada usuario, la plantilla correspondiente a cada mes y a cada usuario sería la siguiente:

```
(deftemplate Consumo
  (slot id )  (slot Fijo)  (slot Movil )  (slot SMS )
); Consumo
```

4.3. Reglas: No se admiten instrucciones if then else en los consecuentes de las reglas. Los planes pueden cambiar sus condiciones (velocidades, precios, etc.) sin que esto implique modificar las reglas.

CAMBIO DE TARIFA
El sistema debe asignar siempre el plan con la tarifa más elevada y nunca permite el cambio a un plan inferior, aunque el usuario lo solicitase. Escribe las reglas necesarias para asignar un plan superior o mostrar un mensaje indicando que no se puede, de acuerdo a las peticiones de los usuarios.

Por ejemplo, si en el sistema se encontraran, además de los hechos iniciales a los 3 planes definidos en el apartado 4.1, los siguientes hechos:

```
(Cliente (id 22) (Plan Cotillea) (Roaming 50))
(Peticion (id 22) (Fibra 50) (TV 0))
```

Debería cambiar el plan del usuario al Plan Ruina, ya que aunque no pida televisión, solicita más velocidad de Internet de la que posee el plan Naufraga, pero si en vez de (Fibra 50) fuera (Fibra 28) debería cambiar al plan Naufraga.

FACTURAR
Para calcular la factura el sistema tiene información detallada del consumo del cliente. Además del precio asignado para cada plan, hay que calcular el importe correspondiente a los minutos excedidos en las llamadas a fijos o móviles, así como el precio de los SMS. Esa información aparecerá directamente en los hechos de tipo Consumo.

5. RETE (1,5 puntos)

Dado la siguiente Base de Reglas y la Base de Hechos, de acuerdo a las plantillas definidas y suponiendo que existe en el sistema una función llamada siguiente_pasillo que devuelve el siguiente pasillo al actual o el primer pasillo cuando se llega al número 11 (esta función no es necesario implementarla ni incluirla en el gráfico):

1. Construye la Red de Redundancia Temporal (RETE).
2. Realiza una simulación de la ejecución correspondiente con la red construida

Plantillas de hechos

```
(deftemplate producto
  (slot id_producto)
  (slot nombre)
  (slot pasillo)
  (slot stock)
  (slot precio)
)
```

```
(deftemplate pedido
  (slot id_cliente)
  (slot id_producto)
  (slot unidades (default 1))
)
```

```
(deftemplate carro
  (slot id_cliente)
  (slot factura (default 0))
  (slot num_productos (default 0))
  (slot pasillo_actual(default 1))
)
```

Base de Reglas

```
(defrule R1_asignar_carro
  (nuevo_cliente ?c1)
  (not (carro (id_cliente ?c1)))
=>
  (assert (carro (id_cliente ?c1) ))
)
```

```
(defrule R2_mover
  (pedido (id_cliente ?c1) (id_producto ?p1))
  ?f1<-(carro (id_cliente ?c1) (pasillo_actual ?pa1))
  (producto (id_producto ?p1) (pasillo ?pa2&~?pa1))
=>
  (modify ?f1  (pasillo_actual (siguientePasillo ?pa1)))
);; mover2
```

```
(defrule R3_comprar
  ?f1<-(producto (pasillo ?pa) (id_producto ?pr) (stock ?s) (precio ?pu))
  ?f2<-(pedido (id_cliente ?c) (id_producto ?pr) (unidades ?u))
  ?f3<-(carro (pasillo_actual ?pa) (id_cliente ?c) (factura ?f) (num_productos ?np))
  (test (>= ?s ?u)) ;; hay cantidad suficiente
=>
  (modify ?f1 (stock (- ?s ?u)))
  (retract ?f2)
(modify ?f3 (num_productos (+ ?np ?u)) (factura (+ ?f  (* ?pu ?u))) )
)
```

Base de Hechos

```
(producto (id_producto 1) (nombre leche) (pasillo 3)  (stock 25) (precio 0.9))
(pedido (id_cliente 33) (id_producto 1) (unidades 4))
(nuevo_cliente  33)
(nuevo_cliente 11)
```