APELLIDOS:	PL:
NOMBRE:	DNI:

ESCUELA DE INGENIERÍA INGORMÁTICA

SISTEMAS INTELIGENTES

Examen Final de Teoría (Búsqueda). Martes 21 de Enero de 2014.

1.- **[3 puntos]** Enunciar la condición de monotonía (o la de consistencia que es equivalente) de los heurísticos, e indicar qué consecuencias tiene esta propiedad en el comportamiento del algoritmo A*.

RESP: Un heurístico h es monótono si cumple que para todo par de nodos del espacio de búsqueda, n1 y n2, $h(n1) \leq h(n2) + c(n1,n2)$; siendo c(n1,n2) el coste de la regla que lleva de n1 a n2. Análogamente, el heurístico es consistente si cumple $h(n1) \leq h(n2) + k(n1,n2)$; siendo k(n1,n2) el coste mínimo para ir de n1 a n2.
Consistencia y monotonía son propiedades equivalentes, es decir h monótono $\Leftrightarrow$ h consistente.
El hecho de que heurístico h tenga estas propiedades tiene unas consecuencias muy interesantes en el comportamiento del algoritmo A*.

a) Si h es consistente entonces h es admisible, y en consecuencia el algoritmo A* encontrará la solución óptima.

b) Si h es consistente, entonces cuando A* elige un nodo n para su expansión, se cumple que $g(n)=g^*(n)$, es decir que en ese momento ya se conoce el mejor camino desde el inicial a n y en consecuencia el algoritmo A* no tiene que rectificar posteriormente este camino. Así, no será necesario reinsertar nodos expandidos en abierta o aplicar la función rectificar lista (dependiendo de la implementación elegida para el proceso de rectificación). Esta es la consecuencia más importante, junto con la admisibilidad, por supuesto.

c) Las condiciones de expansión de los nodos se pueden expresar como: si $g^*(n)+h(n)<C^*$ el nodo n se expande seguro (condición suficiente), si $g^*(n)+h(n) \leq C^*$ el nodo n puede expandirse o no (condición necesaria), si $g^*(n)+h(n)>C^*$ el nodo n no se expandirá (es una condición suficiente de no expansión).

d) Si h es consistente, entonces la secuencia de valores f(n) para los nodos expandidos es no decreciente.

e) La consistencia también tiene una consecuencia importante en la comparación de heurísticos. Si tenemos dos heurísticos consistentes, h1 y h2, tales que $h1(n) \leq h2(n)$ para todo nodo n, entonces todo nodo expandido por h2 y no por h1 cumple $h1(n)=h2(n)=g^*(n)-C^*$, lo cual en términos estadísticos es un suceso poco probable y en consecuencia es esperable que no haya muchos nodos en esta situación, con lo cual en el peor de los casos el número de nodos expandidos por h2 no será mucho mayor que el número de nodos expandidos por h1. Esto es lo que se denomina comparación no estricta.

2.- Consideremos dos permutaciones de los números naturales 1,...,N. Un ciclo se define como un subconjunto de dos o más posiciones (números de 1,...,N) en las que las dos permutaciones tienen los mismos valores, pero en un orden distinto, y de forma que si se prescinde de una o varias posiciones, el subconjunto resultante ya no es un ciclo.

Ejemplo:
C1 = (1 2 3 4 5 6 7 8)
C2 = (1 5 6 7 8 4 3 2)

Las posiciones 2,5,8 y 3,4,6,7 forman sendos ciclos.

Ahora vamos a considerar un esquema de codificación, para un algoritmo genético, mediante permutaciones de enteros 1,...,N, y un operador de cruce que genera un hijo a partir de dos padres y que consiste en los siguientes pasos:

- Los elementos que aparecen en la misma posición en los dos padres, se colocan en el hijo en esa misma posición.
- Se calculan todos los ciclos a partir de las permutaciones de los dos cromosomas, sea S el conjunto de todos estos ciclos.
- Se recorre S, y para cada elemento $s \in S$ se ponen en las posiciones correspondientes del hijo los valores que hay en uno de los padres en esas mismas posiciones. El padre se elige aleatoriamente.

En este ejercicio se pide lo siguiente:

i. **[1 punto]** Indicar los hijos posibles de los cromosomas correspondientes a las permutaciones C1 y C2 anteriores.

C1 = (1 2 3 4 5 6 7 8)
C2 = (1 5 6 7 8 4 3 2)
RESP: Dado que hay dos ciclos distintos, hay (2 padres) × (2 ciclos) = 4 posibles combinaciones que dan lugar a los 4 posibles hijos siguientes
H1 = (1 2 3 4 5 6 7 8), si se eligen los valores de los dos ciclos del padre C1
H2 = (1 5 6 7 8 4 3 2), si se eligen los valores de los dos ciclos del padre C2
H3 = (1 2 6 7 5 4 3 8), si se eligen los valores del primer ciclo de C1 y del segundo de C2
H4 = (1 5 3 4 8 6 7 2), si se eligen los valores del primer ciclo de C2 y del segundo de C1

ii. **[1 punto]** Discutir, de forma breve y razonada, las características de este operador de cruce, y su posible adecuación para los problemas TSP y QAP.

RESP: Este operador de cruce tiene una serie de características evidentes

a) Para cualquier hijo, el valor de un gen siempre coincide con el valor del mismo gen en uno de los padres. En consecuencia, si un valor x no aparece en la posición p de ninguno de los cromosomas de una población, es imposible generar un cromosoma con el valor x en la posición p mediante cruces de cromosomas. Para eso sería necesario utilizar algún operador de mutación.

b) Cada uno de los padres es un potencial hijo, y en algunos casos los padres son los únicos hijos posibles (cuando solamente hay un ciclo).

c) El coste computacional de este operador de cruce es mayor que el coste de los operadores clásicos de cruce en un punto o en dos puntos.

d) Este operador de cruce traslada valores en las mismas posiciones de padres a hijos, pero estos valores pueden no estar en posiciones consecutivas (por ejemplo los del primer ciclo del ejemplo anterior). En consecuencia, podría ser un operador razonable para el QAP, ya que el coste debido a los productos de flujos por distancias entre los elementos de cada ciclo se transfiere directamente de padres a hijos. Sin embargo, no parece un operador muy adecuado para el TSP, ya que si un ciclo no contiene muchas posiciones consecutivas, no hay un coste parcial que se transfiera de un padre a un hijo.

3.- **[2,5 puntos]** Representar el siguiente enunciado en lógica de predicados:

En general los hombres son mortales. Sin embargo, los filósofos, que son hombres, pueden ser mortales o bien inmortales. Sócrates es hombre y no es mortal.

Luego responder a las preguntas siguientes utilizando refutación:

¿Es filósofo Sócrates? y ¿Hay algún filósofo mortal?

RESP: El enunciado se puede modelar de la siguiente forma:

```
public static FOLKnowledgeBase createFilosofosKnowledgeBase(
                    InferenceProcedure infp) {
              FOLKnowledgeBase kb = new FOLKnowledgeBase(DomainFactory.filosofos(),
                          infp);
              //Todas las personas que no son filosofos son mortales
              kb.tell("FORALL x (Persona(x) AND NOT Filosofo(x) => Mortal(x))");
              //Y entre los filosofos los hay mortales
              kb.tell("EXISTS x (Filosofo(x) AND Mortal(x))");
              //y no mortales
              kb.tell("EXISTS x (Filosofo(x) AND NOT Mortal(x))");
              // Socrates es una persona y es mortal
              kb.tell("(Persona(Socrates) AND NOT Mortal(Socrates))");
              return kb;
        }
```

Las dos preguntas se formalizan así, respectivamente:

```
String query = "EXISTS x (Filosofo(x) AND Mortal(x))";
String query = "(Filosofo(Socrates)) ";
```

Y el resultado de la refutación, sin utilizar el mecanismo de Green (predicado Answer), es el siguiente. En los dos casos se llega a la clausula vacía, luego el resultado es afirmativo:

```
-------------------------------------
TFM Resolution,filosofosDemo 1
-------------------------------------

Filosofos Knowledge Base:
FORALL x ((Persona(x) AND NOT(Filosofo(x))) => Mortal(x))
EXISTS x (Filosofo(x) AND Mortal(x))
EXISTS x (Filosofo(x) AND NOT(Mortal(x)))
(Persona(Socrates) AND NOT(Mortal(Socrates)))

Query: EXISTS x (Filosofo(x) AND Mortal(x))
Proof, Answer Bindings: {x=null}
------------------------------------------------------------------------------
|Step | Proof                                  |Justification                |
------------------------------------------------------------------------------
|1    | EXISTS x (Filosofo(x) AND Mortal(x)) |Premise                      |
|2    | EXISTS x (Filosofo(x) AND Mortal(x)) |Premise                      |
|3    | [~Filosofo(v2), ~Mortal(v2)]         |Goal                         |
|4    | [Filosofo(SC0)]                      |Clausified 1                 |
|5    | [Mortal(SC0)]                        |Clausified 2                 |
|6    | [~Filosofo(SC0)]                     |Resolution: 3,5 {v2=SC0}, {} |
|7    | []                                   |Resolution: 4,6 {}, {}       |
------------------------------------------------------------------------------



-------------------------------------
TFM Resolution,filosofosDemo 1
-------------------------------------

Filosofos Knowledge Base:
FORALL x ((Persona(x) AND NOT(Filosofo(x))) => Mortal(x))
EXISTS x (Filosofo(x) AND Mortal(x))
EXISTS x (Filosofo(x) AND NOT(Mortal(x)))
(Persona(Socrates) AND NOT(Mortal(Socrates)))

Query: (Filosofo(Socrates))
Proof, Answer Bindings: {}
```

```
-------------------------------------------------------------------------------
---------
|Step | Proof                                                        |Justification
|
-------------------------------------------------------------------------------
---------
|1    | (Persona(Socrates) AND NOT(Mortal(Socrates)))                |Premise
|
|2    | (Persona(Socrates) AND NOT(Mortal(Socrates)))                |Premise
|
|3    | FORALL x ((Persona(x) AND NOT(Filosofo(x))) => Mortal(x))    |Premise
|
|4    | [~Filosofo(Socrates)]                                        |Goal
|
|5    | [Persona(Socrates)]                                          |Clausified 1
|
|6    | [~Mortal(Socrates)]                                          |Clausified 2
|
|7    | [~Persona(x), Filosofo(x), Mortal(x)]                        |Clausified 3
|
|8    | [~Persona(v0), Filosofo(v0), Mortal(v0)]                     |Renaming of 7
|
|9    | [~Persona(v1), Filosofo(v1), Mortal(v1)]                     |Renaming of 8
|
|10   | [~Persona(Socrates), Mortal(Socrates)]                       |Resolution: 4,9
{}, {}  |
|11   | [~Persona(Socrates)]                                         |Resolution: 6,10
{}, {} |
|12   | []                                                           |Resolution: 5,11
{}, {} |
-------------------------------------------------------------------------------
---------
```

4.- Considerar la función lógica de tres variables definida por la siguiente tabla de verdad

X	Y	Z	F(X,Y,Z)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Se trata de calcular un árbol de decisión que clasifique correctamente las 8 entradas. Concretamente se pide:

a) **[1,5 puntos]** De acuerdo con el heurístico de la máxima ganancia de información, indicar cuál será el atributo elegido para etiquetar el nodo raíz del árbol de decisión.

RESP: Para calcular la máxima ganancia entre los atributos X, Y y Z, comenzamos por calcular la entropía en el nodo raíz. Si etiquetamos los ejemplos de entrenamiento con 1,2,3,4,5,6,7,8, siguiendo el orden de la tabla anterior, en el nodo raíz tendremos que los ejemplos 1,3,4,5,7,8 son del la clase 0 y los ejemplos 2,6 de la clase 1. Así la entropía se calcula como:

$$E(Raíz) = - (6/8)*\log_2(6/8) – (2/8)*\log_2(2/8) = 0.82$$

Si elegimos el atributo X, tendremos dos nodos sucesores x0, con ejemplos 1,2,3,4, y x1, con ejemplos 5,6,7,8. La entropía en el nodo x0 sería (1,3,4 clase 0; 2 clase 1):

$$E(x0) = - (3/4)*\log_2(3/4) – (1/4)*\log_2(1/4) = 0.82$$

La entropía en el nodo x1 es la misma que en x0, E(x1) = 0.82, y la misma que en el nodo raíz, ya que tenemos el mismo porcentaje de ejemplos positivos y negativos.
Luego la ganancia asociada al atributo X será:

$$G(X) = E(Raíz) – Resto(X) = 0.82 – 0.82 = 0$$

Resto(X) = (4/8)*E(x0) + (4/8)*E(x1) = 0.82

La ganancia que tendríamos si elegimos el atributo X es 0. Esto indica que tendríamos la misma incertidumbre que en el nodo raíz.

Si calculamos la ganancia para los atributos Y y Z, tendremos

G(Y) = G(Z) = 0.82 – 0.5 = 0.32

Es positiva e igual para los dos atributos. Luego en este caso, de acuerdo con el heurístico de máxima ganancia, los candidatos para el nodo raíz son los atributos Y y Z. Como hay empate, se elige aleatoriamente.

b) **[1 punto]** Colocar en el nodo raíz el atributo elegido en a) y luego construir el resto del árbol de decisión utilizando cualquier otro criterio para elegir el atributo de cada nodo (por ejemplo el que elige aleatoriamente el atributo). El árbol resultante debe clasificar correctamente los 8 ejemplos de la tabla anterior.

RESP: Elegimos, por ejemplo, el atributo Y para el nodo raíz, y luego el resto con cualquier criterio. Así, un árbol de decisión posible, aunque no el mejor, sería el siguiente: