Inteligencia Artificial II (ejercicio 1) **(21–Junio–2005)**

Apellidos: .....

Nombre: .....

Ejercicio 1: [1.75 puntos]

Contestar las siguientes cuestiones de manera clara y concisa, usando para ello el espacio en blanco que aparece a continuación de cada una de ellas:

(a) ¿Cuál es el sesgo inductivo (y de qué tipo) de los algoritmos FIND-S y Eliminación de Candidatos, respectivamente?

.....

Solución:

- Ambos algoritmos podrían tener sesgo en el lenguaje, si el espacio de hipótesis con el que se trabajara excluyera conceptos.
- El algoritmo FIND-S tiene también sesgo preferencial, ya que durante el proceso de búsqueda de hipótesis consistentes con los ejemplos, toma siempre las hipótesis más específicas.
- El algoritmo de Eliminación de Candidatos no tiene sesgo preferencial, ya que en cada paso va manteniendo todas las hipótesis consistentes con los ejemplos que ha analizado hasta el momento (en realidad, las fronteras específica y general del espacio de versiones).

(b) Describir (en pseudocódigo) el algoritmo de aprendizaje de reglas por cobertura.

.....

Solución:

```
Aprendizaje-por-Cobertura(D,Atributo,v)

1. Hacer Reglas-aprendidas igual a vacío
2. Hacer E igual a D
3. Mientras E contenga ejemplos cuyo valor de Atributo es v, hacer:
   3.1 Crear una regla R sin condiciones y conclusión Atributo=v
   3.2 Mientras que haya en E ejemplos cubiertos por R incorrectamente
        o no queden atributos que usar, hacer:
       3.2.1 Elegir la MEJOR condición A=w para añadir a R, donde
             A es un atributo que no aparece en R y w es un valor de los
             posibles que puede tomar A
       3.2.2 Actualizar R añadiendo la condición A=w a R
   3.3 Incluir R en  Reglas-aprendidas
   3.4 Actualizar E quitando los ejemplos cubiertos por R
4. Devolver Reglas-Aprendidas
```

(c) Supongamos dado el siguiente conjunto de reglas de una base de conocimiento:

```
r1 # si b y c entonces a.
r2 # si d o e o f entonces b.
r3 # si h entonces d.
r4 # si m y n entonces e.
r5 # si l entonces f.
r6 # si j y k entonces c.
f1 hecho j.
f2 hecho k.
preguntable(h).
preguntable(l).
preguntable(n).
```

Supongamos que le preguntamos al sistema si `a` se deduce de la base de conocimiento. Supongamos también que durante el proceso de deducción el sistema realiza dos preguntas al usuario: a la primera de ellas el usuario responde negativamente y a la segunda afirmativamente ¿Cuál es la traza asociada a la segunda de las preguntas? ¿Para qué sirve la traza? ¿Respondería finalmente el sistema que `a` efectivamente se deduce de la base de conocimiento? En caso afirmativo, ¿cuál es el árbol de prueba que justifica tal respuesta?

.....

Solución:
Lo que sigue reproduce la correspondiente sesión con el SBC:

```
2 ?- se_deduce(a).

¿Es cierto h?
Posibles respuestas: [si, no] (seguidas de un punto): no.

¿Es cierto l?
Posibles respuestas: [si, no] (seguidas de un punto): si.
****** Se ha deducido la siguiente respuesta:

      por usted, sabemos que l,
    luego, segun r5 se concluye que f,
  luego, segun r2 se concluye que b,
  y
    por f1, sabemos que j,
    y
    por f2, sabemos que k,
  luego, segun r6 se concluye que c,
luego, segun r1 se concluye que a,
```

Respuestas:

- La traza es la lista de reglas que conectan el objetivo original, con lo que se le está preguntando al usuario. En este caso, la traza asociada a la segunda pregunta ("¿es cierto `l`?") es la lista de reglas `[r5,r2,r1]`.
- La traza sirve para gestionar la respuesta del sistema a una posible pregunta "¿por qué?" por parte del usuario.
- Como se observa, el sistema responde afirmativamente.

- El árbol de prueba se obtiene directamente de la justificación anterior presentada por el sistema.

Observaciones:

- Nótese que `n`, aunque es preguntable, nunca llega a preguntarse: esto es debido a que `m` no está en la base de conocimiento y por tanto la regla `r4` no llega a usarse.
- No debe confundirse el árbol de prueba, que refleja la deducción que finalmente se ha encontrado, con el proceso de búsqueda de la prueba (que también se puede representar en forma de árbol). Aquí se está pidiendo el árbol de prueba.

Inteligencia Artificial II (ejercicio 2) **(21–Junio–2005)**

Apellidos: .....

Nombre: .....

Ejercicio 2 [1.75 puntos]

Se considera el siguiente programa CLIPS:

```
(defmodule MAIN
  (export deftemplate lista elemento))

(deffacts MAIN::inicial
  (elemento 5)
  (lista 5 6 7 6 5 2 4))

(defrule MAIN::elimina
  (elemento ?e)
  ?l <- (lista $?i ?e $?f)
  =>
  (assert (lista ?i ?f))
  (retract ?l))

(defrule MAIN::final
  (declare (salience -10))
  =>
  (focus OTRO))

(defmodule OTRO
  (import MAIN deftemplate lista elemento))

(defrule OTRO::cambia
  ?l <- (lista $?i ?e1 $?m ?e2&:(< ?e1 ?e2) $?f)
  =>
  (assert (lista ?i ?e2 ?m ?e1 ?f))
  (retract ?l))
```

1. Rellenar la tabla de seguimiento que se adjunta. Se tienen que incluir TODAS las activaciones de las reglas, el valor que toma la variable `?i` en cada de las activaciones de la regla `elimina` y los valores de las variables `?e1` y `?e2` en cada una de las activaciones de la regla `cambia`.

   .....

   Solución:

Hechos	E	S	Agenda MAIN	D	S	Agenda OTRO	D	S
f0 (initial-fact)	0		-10 final f0	3				
f1 (elemento 5)	0							
f2 (lista 5 6 7 6 5 2 4)	0	1	0 elimina f1,f2:(5 6 7 6)	-	1	cambia f2:5,6	-	1
			0 elimina f1,f2:()	1		cambia f2:5,7	-	1
						cambia f2:6,7	-	1
						cambia f2:5,6	-	1
						cambia f2:2,4	-	1
f3 (lista 6 7 6 5 2 4)	1	2	0 elimina f1,f3:(6 7 6)	2		cambia f3:2,4	-	2
						cambia f3:6,7	-	2
f4 (lista 6 7 6 2 4)	2	4				cambia f4:2,4	-	4
						cambia f4:6,7	4	
f5 (lista 7 6 6 2 4)	4	5				cambia f5:2,4	5	
f6 (lista 7 6 6 4 2)	5							

2. Indicar los hechos que quedan finalmente en memoria.

   .....

   Solución:

   `f0: (initial-fact), f1: (elemento 5), f6: (lista 7 6 6 4 2)`

3. Describir brevemente el significado del programa anterior ¿qué papel juega el módulo OTRO?

   .....

   Solución:

   Elimina del hecho `lista` las apariciones del componente que aparece en el hecho `elemento` y ordena `lista` (orden decreciente). El módulo OTRO se encarga de la ordenación.

4. Mostrar la evolución de la pila de focos.

   .....

   Solución:

   

5. Escribir un programa, con los mismos efectos que el anterior, utilizando únicamente dos reglas (sin módulos, funciones o prioridades).

   .....

   Solución:

```
(defrule elimina
  (elemento ?e)
  ?l <- (lista $?i ?e $?f)
  =>
  (assert (lista ?i ?f))
  (retract ?l))

(defrule cambia
  (elemento ?e)
  (not (lista $? ?e $?))
  ?l <- (lista $?i ?e1 $?m ?e2&:(< ?e1 ?e2) $?f)
  =>
  (assert (lista ?i ?e2 ?m ?e1 ?f))
  (retract ?l))
```

Inteligencia Artificial II (ejercicio 3) **(21–Junio–2005)**

Apellidos: .....

Nombre: .....

Ejercicio 3 [1.75 puntos]

La siguiente tabla muestra las preferencias de una familia a la hora de decidir la compra de una segunda vivienda en el mercado de viviendas usadas, en función de determinadas características: si está en el campo o en la playa, la distancia a la primera vivienda, el color de la fachada, la década en que fué construida y si se trata de un piso o una casa:

Ej.	Situación	Lejanía	Color	Década	Tipo	Comprar
E_1	playa	cerca	blanco	ochenta	piso	si
E_2	playa	medio	verde	setenta	casa	no
E_3	playa	medio	blanco	noventa	piso	si
E_4	campo	lejos	rojo	ochenta	piso	no
E_5	playa	cerca	amarillo	ochenta	piso	si

Suponiendo que el espacio de hipótesis es el de aquellas expresables mediante conjunción de restricciones sobre los valores de los atributos, aplicar (detallando cada uno de los pasos realizados) el algoritmo de eliminación de candidatos usando como entrada la tabla anterior. Según lo aprendido ¿qué tipo de vivienda se está intentando comprar?

.....

Solución:

Inicialmente, las cotas general y específica son las siguientes:

Cota general: `G = {<?, ?, ?, ?, ?>}`

Cota específica: `S = {<∅,∅,∅,∅,∅>}`

Para cada elemento del conjunto `D`:

1. `<playa,cerca,blanco,ochenta,piso>`$^+$

   Eliminamos de `G` cualquier hipótesis inconsistente (no hay):

   `G = {<?, ?, ?, ?, ?>}`.

   Para cada hipótesis de `S` inconsistente:

   * `<∅,∅,∅,∅,∅>`

     La eliminamos de `S`: `S = ∅`.

     Generalizaciones minimales de `<∅,∅,∅,∅,∅>` consistentes con el ejemplo:

     * `<playa,cerca,blanco,ochenta,piso>`

       Es más específica que `<?, ?, ?, ?, ?>`.

     `S = {<playa,cerca,blanco,ochenta,piso>}`.

   Eliminamos de `S` aquellas hipótesis para las que existe en `S` otra más específica (no hay):

   `S = {<playa,cerca,blanco,ochenta,piso>}`.

2. <playa,medio,verde,setenta,casa>$^-$

   Eliminamos de S cualquier hipótesis inconsistente (no hay):

   S = {<playa,cerca,blanco,ochenta,piso>}.

   Para cada hipótesis de G inconsistente:

   * <?, ?, ?, ?, ?>

     La eliminamos de G: G = $\emptyset$.

     Especializaciones minimales de <?, ?, ?, ?, ?> consistentes con el ejemplo:

     * <campo,?,?,?,?>, <?,lejos,?,?,?>, <?,?,amarillo,?,?>, <?,?,rojo,?,?>, <?,?,?,noventa, ?>, (no son más generales que alguna de S)
     * <?,cerca,?,?,?>, <?,?,blanco,?,?>, <?,?,?,ochenta,?>, <?,?,?,?,piso> son más generales que alguna de S.

   G = {<?,cerca,?,?,?>, <?,?,blanco,?,?>,<?,?,?,ochenta,?>, <?,?,?,?,piso>}.

   Eliminamos de G aquellas hipótesis para las que existe en G otra más general (no hay):

   G = {<?,cerca,?,?,?>, <?,?,blanco,?,?>,<?,?,?,ochenta,?>, <?,?,?,?,piso>}.

3. <playa,medio,blanco,noventa,piso>$^+$

   Eliminamos de G cualquier hipótesis inconsistente:

   G = {<?,?,blanco,?,?>, <?,?,?,?,piso>}.

   Para cada hipótesis de S inconsistente:

   * <playa,cerca,blanco,ochenta,piso>

     La eliminamos de S: S = $\emptyset$.

     Generalizaciones minimales de <playa,cerca,blanco,ochenta,piso> consistentes con el ejemplo:

     * <playa,?,blanco,?,piso>

       Es más específica que alguna de G.

   S = {<playa,?,blanco,?,piso>}.

   Eliminamos de S aquellas hipótesis para las que existe en S otra más específica (no hay):

   S = {<playa,?,blanco,?,piso>}.

4. <campo,lejos,rojo,ochenta,piso>$^-$

   Eliminamos de S cualquier hipótesis inconsistente (no hay):

   S = {<playa,?,blanco,?,piso>}.

   Para cada hipótesis de G inconsistente:

   * <?, ?, ?, ?, piso>

     La eliminamos de G: G = {<?,?,blanco,?,?>}.

     Especializaciones minimales de <?, ?, ?, ?, piso> consistentes con el ejemplo y más generales que alguna de S: <playa,?,?,?,piso>, <?,?,blanco,?,piso>.

     G = {<?,?,blanco,?,?>, <playa,?,?,?,piso>, <?,?,blanco,?,piso>}.

     Eliminamos de G aquellas hipótesis para las que existe en G otra más general:

     G = {<?,?,blanco,?,?>, <playa,?,?,?,piso>}.

* <playa,cerca,amarillo,ochenta,piso>$^+$

  Eliminamos de G cualquier hipótesis inconsistente:

  G = {<playa,?,?,?,piso>}.

  Para cada hipótesis de S inconsistente:

  * <playa,?,blanco,?,piso>
    La eliminamos de S: S = $\emptyset$.
    Generalizaciones minimales de <playa,?,blanco,?,piso> consistentes con el ejemplo y más específica que alguna de G: <playa,?,?,?,piso>.

  S = {<playa,?,?,?,piso>}.

  Eliminamos de S aquellas hipótesis para las que existe en S otra más específica (no hay):

  S = {<playa,?,?,?,piso>}.

Finalmente, se obtiene G = {<playa,?,?,?,piso>} y S = {<playa,?,?,?,piso>}. Por tanto, se intenta comprar un piso en la playa.

Inteligencia Artificial II (ejercicio 4) **(21–Junio–2005)**

Apellidos: .....

Nombre: .....

Ejercicio 4 [1.75 puntos]

La frase en lenguaje natural *"Dos informáticos son andaluces"* tiene el siguiente significado:

$$\exists x \exists y [x \neq y \wedge informatico(x) \wedge informatico(y) \wedge andaluz(x) \wedge andaluz(y)]$$

Definir, de manera similar a la GCD que se ha dado en clase para frases que empiezan con el determinante "algún", una GCD para reconocer y obtener el significado de frases que empiezan con el determinante "dos".

Por simplificar, como léxico para los nombres de propiedades usar simplemente "informáticos", "europeos" y "andaluces". Además, no es necesario contemplar el uso de verbos transitivos (tan solo frases con el verbo copulativo "son"). Lo que sigue es un ejemplo de análisis de frases con la gramática que se pide:

```
?- phrase(oración_dos(S),[dos,informáticos,son,andaluces]).

S = existe(X, existe(Y, no(X=Y) y (informático(X) y informático(Y))
                        y andaluz(X) y andaluz(Y)))

?- phrase(oración_dos(S),[dos,europeos,son,andaluces]).

S = existe(X, existe(Y, no(X=Y) y (europeo(X) y europeo(Y))
                        y andaluz(X) y andaluz(Y)))
```

Indicación: La principal modificación, respecto de las reglas que reconocen frases con "algún", radica en que hay que usar dos variables para la cuantificación existencial, aumentando por tanto el número de argumentos de algunas categorías sintácticas.

.....

Solución:

Lo que sigue es una posible solución. Nótese el papel del determinante dentro de la semántica composicional: su significado se puede ver como una función que a partir de los significados de los distintos componentes de la frase obtiene el significado de la sentencia completa.

```
:-op(900,xfy,y).
:-op(500,fx,no).

oración_dos(S) --> sujeto_det(X,Y,SSV,S), sintagma_verbal(X,Y,SSV).
sujeto_det(X,Y,SSV,S) --> determinante(X,Y,Prop,SSV,S),nombre_propiedad(X,Y,Prop).
determinante(X,Y,Prop,SSV,existe(X,existe(Y, no (X = Y) y Prop y SSV ))) --> [dos].
sintagma_verbal(X,Y,SV) --> verbo_cop,nombre_propiedad(X,Y,SV).
verbo_cop --> [son].
nombre_propiedad(X,Y,informático(X) y informático(Y)) --> [informáticos].
nombre_propiedad(X,Y,andaluz(X) y andaluz(Y)) --> [andaluces].
nombre_propiedad(X,Y,europeo(X) y europeo(Y)) --> [europeos].
```