Inteligencia Artificial II **(18–Junio–1999)**

Apellidos:
Nombre:

Ejercicio 1 [1.5 puntos] El procedimiento `(es-consecuencia S F)` devuelve `t` si la fórmula `F` es consecuencia del conjunto de fórmulas `S` y devuelve `nil` en caso contrario. Se considera la siguiente definición del dicho procedimiento.

```
(defun es-consecuencia (S F)
  (every #'(lambda (I) (if (es-modelo-conjunto I S)
                           (es-modelo-formula I F)
                           t))
         (interpretaciones-conjunto S)))
```

Explicar si la definición anterior es correcta o incorrecta y, en el caso de ser incorrecta,

1. dar un ejemplo de un conjunto `S` y una fórmula `F` tales que `F` no sea consecuencia de `S` y el valor de `(es-consecuencia S F)` sea `t`.
2. decir lo que hay que añadirle a la definición para que sea correcta

---

Ejercicio 2 [1.5 puntos] En el método de los tableros semánticos se usa el procedimiento `(tiene-tablero-cerrado S)` para determinar si el conjunto de fórmulas `S` tiene un tablero cerrado. Se considera la siguiente definición del dicho procedimiento.

```
(defun tiene-tablero-cerrado (S)
  (cond ((setf F (find-if #'es-formula-alfa S))
         (tiene-tablero-cerrado (append (componentes F) (remove F S))))
        ((setf F (find-if #'es-formula-beta S))
         (let* ((componentes (componentes F))
                (F1 (first componentes))
                (F2 (second componentes))
                (temp (remove F S)))
           (and (tiene-tablero-cerrado (cons F1 temp))
                (tiene-tablero-cerrado (cons F2 temp)))))
        (t nil)))
```

Explicar si la definición anterior es correcta o incorrecta y, en el caso de ser incorrecta,

1. dar un ejemplo de un conjunto `S` que tenga tablero cerrado y el valor de `(tiene-tablero-cerrado S)` sea `nil`
2. decir lo que hay que añadirle a la definición para que sea correcta

---

Ejercicio 3 [2.5 puntos] El procedimiento `(prueba-consecuencia S F)` prueba que la fórmula `F` es consecuencia del conjunto de fórmulas `S` utilizando resolución con eliminaciones. Aplicando dicho procedimiento, demostrar que

$$\{\neg q \to p \vee r, p \to r\} \models q \vee r$$

[**Nota:** *Indicar el soporte, la búsqueda de la prueba y la prueba obtenida*]

Ejercicio 4 [2 puntos]

1. Definir en Lisp el procedimiento `(unifica e1 e2)` que devuelva un unificador de máxima generalidad de las expresiones `e1` y `e2`, si son unificables y devuelva `fallo` en caso contrario.

   [**Nota:** *Los procedimientos* **auxiliares** *estudiados en el curso basta describirlos en lenguaje natural.*]

2. Aplicar el procedimiento `unifica` para decidir si los siguientes pares de expresiones son unificables y calcular, en su caso, un unificador de máxima generalidad

   - `(p (f X) Z)` con `(p Y (g Y))`
   - `(p (f X) X)` con `(p Y (g Y))`

Ejercicio 5 [2.5 puntos]

1. Se considera el siguiente base de conocimiento

   ```
   gusta(X,Y) <- gato(X), pez(Y).
   pez(X) <- atún(X).
   atún(a1) <-.
   atún(a2) <-.
   gato(g1) <-.
   ```

   Construir el árbol SLD correspondiente a dicha base y el objetivo

   ```
   <- gusta(X,Y).
   ```

   indicando las respuestas obtenidas.

2. Se considera el siguiente base de conocimiento en CLIPS

   ```
   (deffacts inicial
     (atun a1)
     (atun a2)
     (gato g1))

   (defrule r1
     (gato ?x)
     (pez ?y)
     =>
     (assert (gusta ?x ?y)))

   (defrule r2
     (atun ?x)
     =>
     (assert (pez ?x)))
   ```

   Escribir la tabla de seguimiento de su ejecución e indicar los hechos que quedan finalmente en memoria.