Inteligencia Artificial I **(1–Febrero–2000)**

Apellidos:
Nombre:

Ejercicio 1 [2.5 puntos]

El análisis sintáctico puede verse como una búsqueda en espacio de estados. Supongamos que deseamos analizar la frase "el niño llora". Para ello se parte del estado inicial `((F) (el niño llora))` y se consideran los operadores correspondientes a la gramática siguiente

```
R1: F   -> SN SV
R2: SN  -> ART N
R3: SN  -> ART N ADJ
R4: SV  -> V SN
R5: SV  -> V
R6: ART -> el
R7: N   -> niño | hombre
R8: V   -> llora | rie
R9: ADJ -> cojo
```

donde `F` representa una frase, `SN` un sintagma nominal, `SV` un sintagma verbal, `ART` un artículo, `N` un nombre, `ADJ` un adjetivo y `V` un verbo.

Los operadores correspondientes a las 5 primeras reglas pueden aplicarse a un estado cuando el primer elemento del estado empieza por la parte izquierda de la regla correspondiente. Al aplicarlo sustituyen la parte izquierda de la regla por su parte derecha en el primer elemento del estado; por ejemplo, al aplicar el operador `R2` al estado `((SN SV) (el niño llora))` se obtiene `((ART N SV) (el niño llora))`.

Los operadores correspondientes a las 4 últimas reglas pueden aplicarse a un estado cuando el primer elemento del estado empieza por la parte izquierda de la regla y el segundo elemento del estado empieza por un elemento de la parte derecha de la regla. Al aplicarlo se eliminan los primeros elementos de los dos elementos del estado; por ejemplo, al aplicar el operador `R7` al estado `((N SV) (niño rie))` se obtiene el estado `((SV) (rie))` y el mismo estado se obtiene al aplicar `R7` al estado `((N SV) (hombre rie))`

Los estados finales son los que tienen la lista vacía como primer y segundo elemento.

1. Escribir la representación en Lisp del problema de análisis sintáctico, de forma que cada operador corresponda a una de las reglas de la gramática.
2. Explicar, indicando los estados analizados y la evolución de `ABIERTOS`, la solución del problema obtenida mediante `(busqueda-en-profundidad)`.

3. ¿Qué hay que modificar en la representación anterior para analizar la frase "el hombre cojo ríe"?. ¿Qué se obtiene en esta caso aplicando `(busqueda-en-profundidad)`?.

Ejercicio 2 [1.5 puntos]

El procedimiento `(verifica plan)` devuelve `t` si el `plan` es una solución del problema y `nil` en caso contrario (mostrando, además, los estados obtenidos y los operadores aplicados). Por ejemplo, en el problema del granjero,

```
> (verifica '(pasan-granjero-y-cabra
              pasa-granjero-solo))
(I I I I) PASAN-GRANJERO-Y-CABRA
(D I D I) PASA-GRANJERO-SOLO
(I I D I) No es estado final
NIL
```

Se considera la siguiente definición de dicho procedimiento.

```
(defun verifica (plan &optional (estado *estado-inicial*))
  (cond ((null plan)
         (cond ((es-estado-final estado)
                (format t "~&~a Estado final~&" estado))
               (t (format t "~&~a No es estado final~&" estado)
                  nil)))
        (t (format t "~&~a ~a" estado (first plan))
           (verifica (rest plan) ((first plan) estado)))))
```

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

1. dar ejemplos explicando los errores de la definición y
2. decir cómo modificar la definición para que sea correcta.

Ejercicio 3 [2 puntos]

Definir el procedimiento recursivo `busqueda-en-profundidad-rec-simple` que busca en profundidad una solución a un problema representado como problema de espacio de estados, pero **sin** detectar caminos cíclicos ni redundantes y **sin** usar `loop`.

[*Indicación*: Este procedimiento debe tener un argumento `abiertos` opcional. Inicialmente, `abiertos` es una lista cuyo único elemento es el nodo inicial. En cada llamada recursiva `abiertos` contiene la lista de nodos pendientes de analizar.]

Ejercicio 4 [1.5 puntos]
Se considera el siguiente programa

```
p(0,X,X).
p(s(X),Y,s(Z)) :- p(X,Y,Z).
```

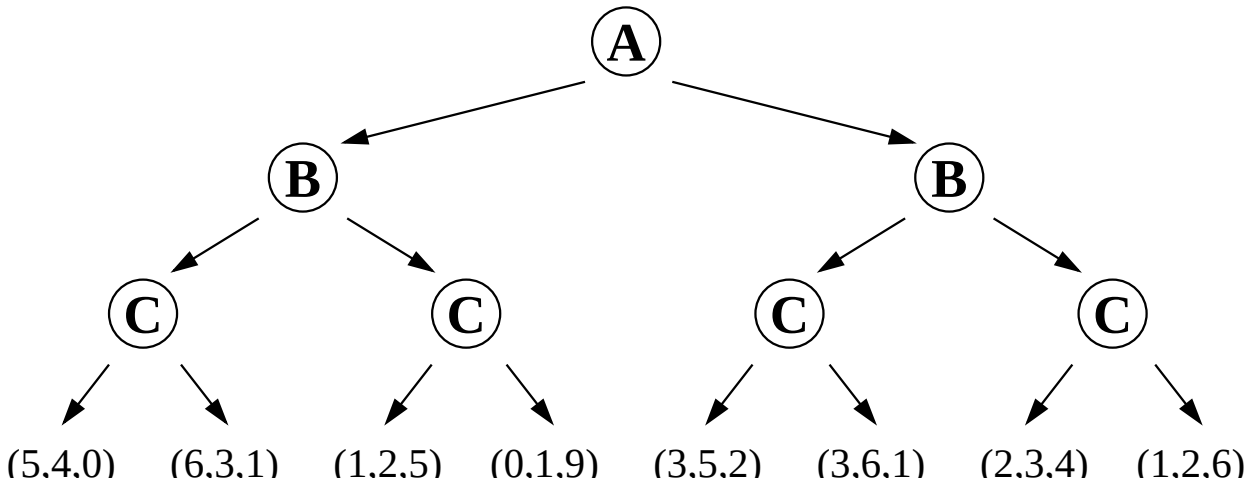
Construir el árbol de resolución SLD correspondiente a dicho programa y a la pregunta

```
?- p(s(X),X,s(s(s(0)))).
```

indicando las respuestas obtenidas.

Ejercicio 5 [2.5 puntos]

Vamos a considerar el siguiente árbol que representa tres jugadas sucesivas de un juego en el que hay 3 jugadores a los que llamaremos A, B y C. Estos jugadores participan alternativamente, siendo el primero el jugador A, el segundo el B y el tercero el C. La función de evaluación estática asociada a los nodos hoja devuelve una lista de tres valores que son respectivamente las ventajas de cada uno de los jugadores en dicho estado. Así, un nodo con el valor (1 2 5), representa un estado en el que el jugador A tiene una ventaja de 1, el jugador B una ventaja de 2 y el jugador C una ventaja de 5.



Determinar qué lista de valores representando las ventajas de los jugadores, han de asignarse a cada uno de los nodos interiores del árbol, siguiendo un proceso similar al del algoritmo Minimax.

La función de evaluación estática asigna a los nodos hoja una lista de tres valores (i j k). Teniendo en cuenta que $0 \leq i, j, k$ y $i + j + k \leq 10$, determinar qué rama del árbol anterior se podría haber evitado analizar y explicar por qué.