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

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 este caso aplicando `(busqueda-en-profundidad)`?.

Solución 1.1

```
(defparameter *estado-inicial*
  '((F) (el niño llora)))

(defun es-estado-final (estado)
  (equal estado '(()())))

(defparameter *operadores*
  '(R1 R2 R3 R4 R5 R6 R7 R8 R9))

(defun R1 (estado)
  (when (eq (first (first estado)) 'F)
        (list (append '(SN SV) (rest (first estado)))
              (second estado))))

(defun r2 (estado)
  (when (eq (first (first estado)) 'SN)
        (list (append '(ART N) (rest (first estado)))
              (second estado))))

(defun r3 (estado)
  (when (eq (first (first estado)) 'SN)
        (list (append '(ART N ADJ) (rest (first estado)))
              (second estado))))

(defun r4 (estado)
  (when (eq (first (first estado)) 'SV)
        (list (append '(V SN) (rest (first estado)))
              (second estado))))

(defun r5 (estado)
  (when (eq (first (first estado)) 'SV)
        (list (append '(V) (rest (first estado)))
              (second estado))))

(defun r6 (estado)
  (when (and (eq (first (first estado)) 'ART)
             (eq (first (second estado)) 'el))
        (list (rest (first estado))
              (rest (second estado)))))
```

```
(defun r7 (estado)
  (when (and (eq (first (first estado)) 'N)
             (member (first (second estado)) '(niño hombre)))
        (list (rest (first estado))
              (rest (second estado)))))

(defun r8 (estado)
  (when (and (eq (first (first estado)) 'V)
             (member (first (second estado)) '(llora rie)))
        (list (rest (first estado))
              (rest (second estado)))))

(defun r9 (estado)
  (when (and (eq (first (first estado)) 'ADJ)
             (eq (first (second estado)) 'COJO))
        (list (rest (first estado))
              (rest (second estado)))))

(defun aplica (operador estado)
  (funcall (symbol-function operador) estado))
```

.....

Solución 1.2

Abiertos	Regla	Solución
(((f) (el niño llora)))	1	
(((sn sv) (el niño llora)))	2,3	1
(((art n sv) (el niño llora)) ((art n sv) (el niño llora)))	6	2
(((n sv) (niño llora)) ((art n sv) (el niño llora)))	7	6
(((sv) (llora)) ((art n sv) (el niño llora)))	4,5	7
(((v sn) (llora)) ((v) (llora)) ((art n sv) (el niño llora)))	8	
(((sn) ()) ((v) (llora)) ((art n sv) (el niño llora)))	2,3	

```
| (((art n) ())                  |       |          |
|  ((art n adj) ())              |       |          |
|  ((v) (llora))                 |       |          |
|  ((art n sv) (el niño llora))) |       |          |
| (((art n adj) ())              |       |          |
|  ((v) (llora))                 |       |          |
|  ((art n sv) (el niño llora))) |       |          |
| (((v) (llora))                 | 8     | 5        |
|  ((art n sv) (el niño llora))) |       |          |
| ((() ())                       | Final | 8        |
|  ((art n sv) (el niño llora))) |       |          |
+--------------------------------+-------+----------+
```

La solución obtenida es `(R1 R2 R6 R7 R5 R8)`.

.....

Solución 1.3

Lo que se necesita cambiar es la definición del estado inicial por la siguiente

```
(setf *estado-inicial* '((f) (el hombre cojo rie)))
```

La solución obtenida es `(R1 R3 R6 R7 R9 R5 R8)`.

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.

. . . . . .

Solución

El primer error consiste en la aplicación de los operadores a los estados. Para arreglarlo, hay que cambiar

```
((first plan) estado)
```

por

```
(aplica (first plan) estado)
```

Arreglado este error la definición sigue siendo incorrecta. Por ejemplo,

```
> (verifica '(pasan-granjero-y-cabra
              pasa-granjero-solo
              pasan-granjero-y-col
              pasan-granjero-y-cabra
              pasan-granjero-y-lobo
              pasa-granjero-solo
              pasan-granjero-y-cabra))
(I I I I) PASAN-GRANJERO-Y-CABRA
(D I D I) PASA-GRANJERO-SOLO
(I I D I) PASAN-GRANJERO-Y-COL
(D I D D) PASAN-GRANJERO-Y-CABRA
(I I I D) PASAN-GRANJERO-Y-LOBO
(D D I D) PASA-GRANJERO-SOLO
(I D I D) PASAN-GRANJERO-Y-CABRA
(D D D D) Estado final
NIL
```

el valor es `NIL` en lugar de `T`. Para corregirlo, hay que cambiar

```
((es-estado-final estado)
 (format t "~&~a Estado final~&" estado))
```

por

```
((es-estado-final estado)
 (format t "~&~a Estado final~&" estado)
 t)
```

Arreglado este error la definición sigue siendo incorrecta. Por ejemplo,

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

La causa es que el operador `pasa-granjero-solo` no es aplicable al estado inicial. Para corregirlo, se añade como primera condición una prueba para ver si el operador es aplicable, quedando la definición como sigue

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

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.]

. . . . . .

Solución

```
(defun busqueda-en-profundidad-rec-simple
  (&optional abiertos (list (crea-nodo :estado *estado-inicial*
                                       :camino nil)))
  (when abiertos
    (let ((actual (first abiertos)))
      (if (es-estado-final (estado actual))
          actual
          (busqueda-en-profundidad-rec-simple
                 (append (sucesores actual)
                         (rest abiertos)))))))
```

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

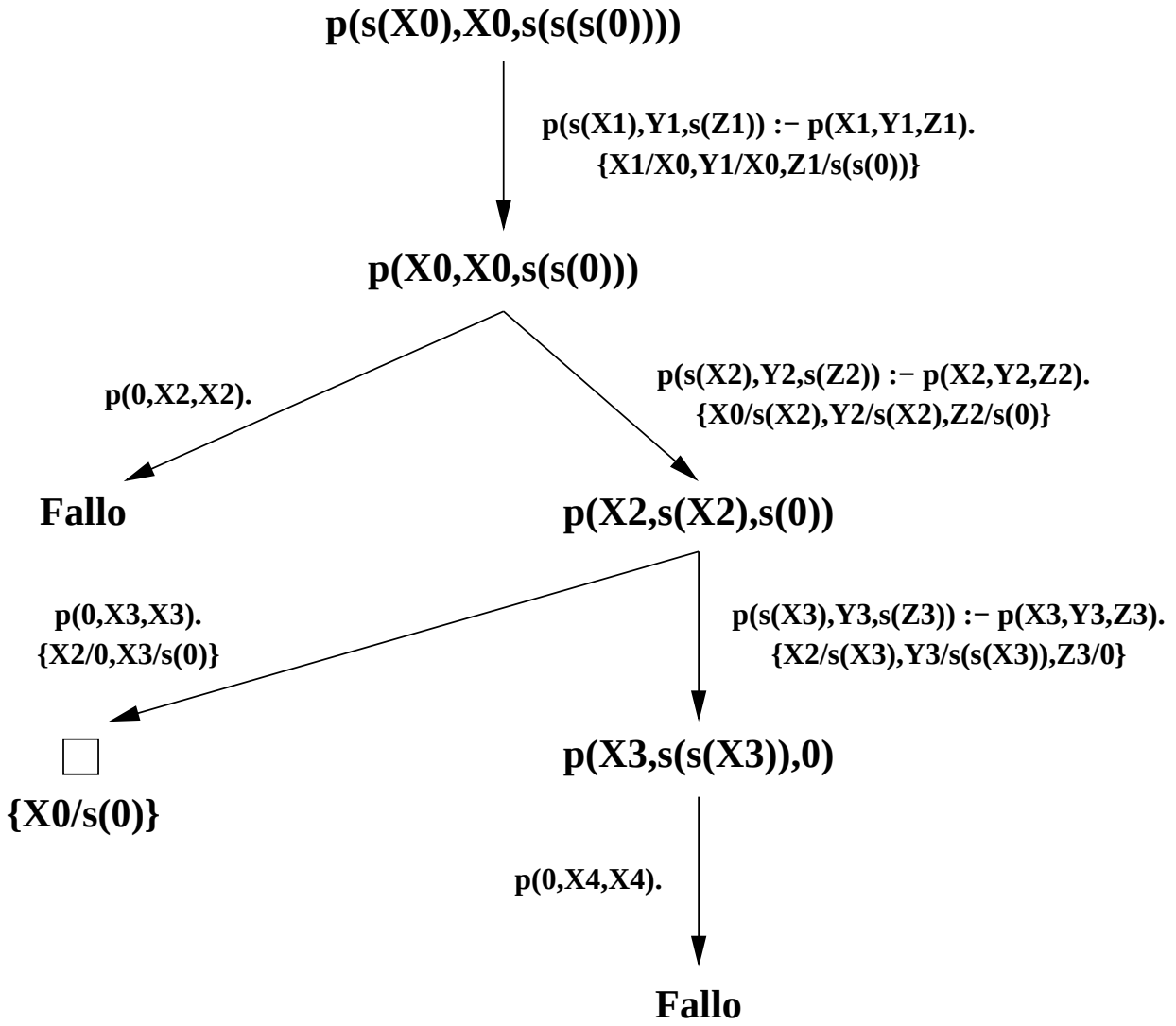
```
?- p(s(X),X,s(s(s(0)))).
```

indicando las respuestas obtenidas.

. . . . . .

Solución

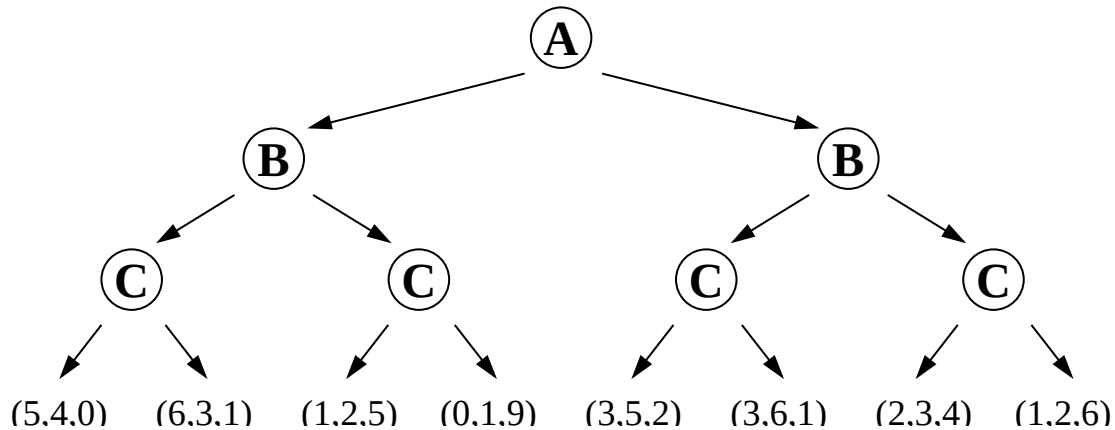
El árbol de resolución SLD pedido es el siguiente:



en el que la única respuesta obtenida es $X/s(0)$.

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.

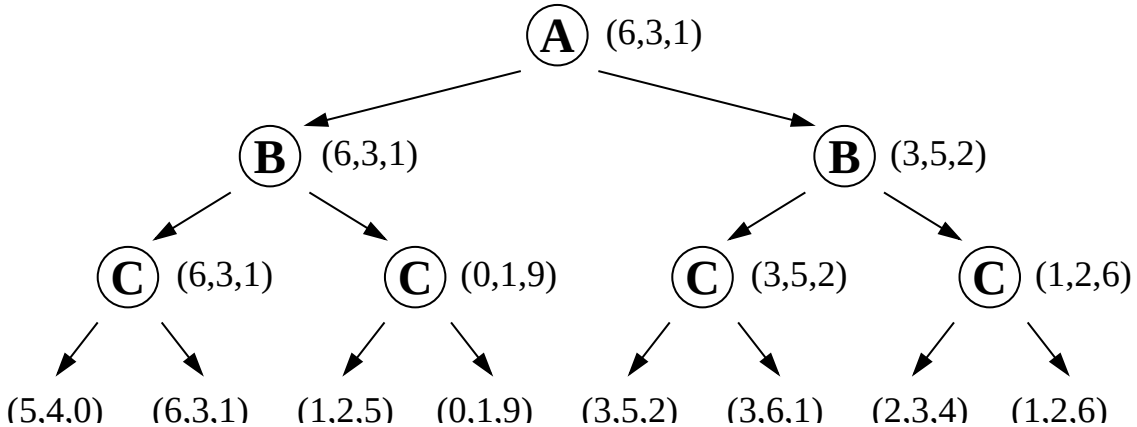
La función de evaluación estática asigna a los nodos hoja una lista de tres valores (i j k). Teniendo en cuenta que $0 \leq i, j, k$ y $i+j+k \leq 10$, determinar qué rama del árbol anterior se podría haber evitado analizar y explicar por qué.

. . . . . .

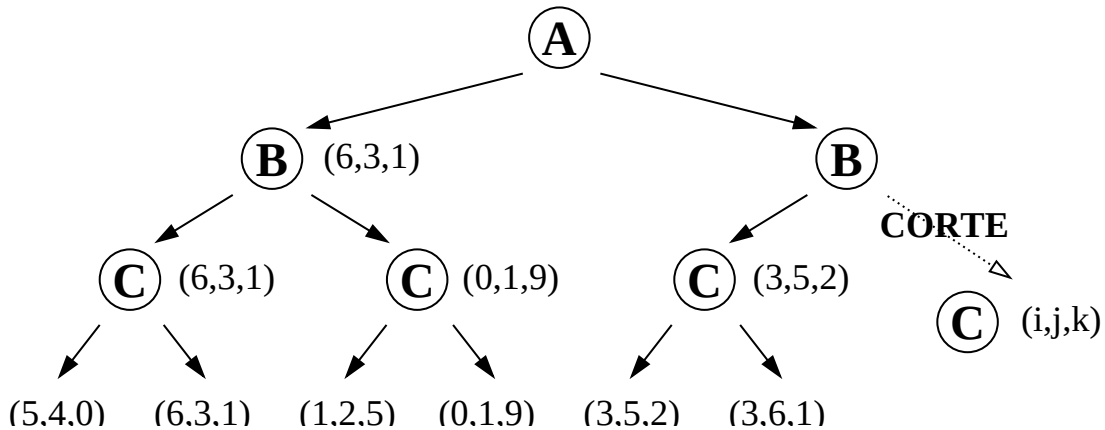
Solución

En el algoritmo Minimax, cada jugador escoge aquella posición que más le favorece. Aquí, de manera análoga, cada jugador escogerá la opción que mayor ventaja le proporcione, así, si le toca jugar a C y sus opciones son dos nodos con valores (5,4,0) y (6,3,1), escogerá el nodo con valor (6,3,1), pues es el que mayor ventaja le proporciona.

El siguiente árbol muestra que valores se asignan a cada uno de los nodos interiores:



Evitaremos el análisis de una rama del árbol cuando podamos concluir que el valor obtenido de su análisis no es importante, sin necesidad de calcularlo explícitamente. Esta situación se da una sóla vez en el árbol anterior, produciéndose un sólo corte, tal y como se muestra a continuación:



Supongamos que el valor del nodo C cortado es (i,j,k). Si $j < 5$, entonces el jugador B escogerá la rama de su izquierda pues le proporciona mayor ventaja, finalmente el jugador A escogerá la rama de la izquierda. Si $j \geq 5$, entonces, como $0 \leq i, j, k$ y $i + j + k \leq 10$, se tendrá que $i < 6$, por tanto, sea cual sea la elección de B, la ventaja de A en ese nodo será menor que 6 y el jugador A escogerá la rama de la izquierda.