Resultados de los ejercicios del Examen de Procesadores de Lenguajes

(Convocatoria Ordinaria) celebrado el 30 de Junio de 2006.

3. (1 punto) Sea la gramática con las producciones siguientes:

Programa	$\longrightarrow$	**block** Sentencias **endblock**
Sentencias	$\longrightarrow$	Sentencias **;** Sentencia
	$\mid$	Sentencia
Sentencia	$\longrightarrow$	ident **=** Exp
	$\mid$	**do** Sentencia **until** Exp
	$\mid$	Programa
Exp	$\longrightarrow$	Exp **+** Exp
	$\mid$	Exp **-** Exp
	$\mid$	Exp ***** Exp
	$\mid$	Exp **/** Exp
	$\mid$	***** Exp
	$\mid$	**&** Exp
	$\mid$	**-** Exp
	$\mid$	const
	$\mid$	ident
const	$\longrightarrow$	const num
	$\mid$	num
num	$\longrightarrow$	$0\|1\|2\|\cdots\|9$
ident	$\longrightarrow$	ident letra
	$\mid$	letra
letra	$\longrightarrow$	$a\|b\|c\|\cdots\|z$

Obtener la tabla de tokens con el máximo nivel de abstracción suponiendo que:

(a) *(0.6 puntos) Se va a realizar traducción:* Las últimas producciones definen directamente secuencias de caracteres que se pueden definir mediante expresiones regulares. Alcanzando su máxima abstracción de acuerdo a su significado, la lista de palabras (secuencias de símbolos con significado propio) son las siguientes:

block, endblock, ;, ident, =, do, until, +, -, *, /, &, const

Teniendo en cuenta la misión sintáctica, sería posible agrupar los siguientes lexemas: **-,*** (son operadores binarios y unarios) y **+,/** (son operadores exclusivamente binarios). La tabla de tokens con el máximo nivel de abstracción queda como sigue:

Token	Patrón	Atributos
BLOCK	"block"	
ENDBLOCK	"endblock"	
PYC	";"	
ID	[a-z]+	
ASIG	"="	
DO	"do"	
UNTIL	"until"	
CONST	[0-9]+	
MENPOR	"−"|"∗"	0: "−" 1: "∗"
MASDIV	"+"|"/"	0: "+" 1: "/"
DIR	"&"	

(b) *(0.4 puntos) Sólo se va a realizar análisis sintáctico:* La gramática de partida obedece a una gramática libre de contexto causado por la iteración de sentencias basadas en la producción `Programa`. Sería posible abstraer hasta la sentencia de asignación (sería posible agrupar lexemas del apartado anterior que quedaban separados atendiendo a su significado). La tabla de tokens con el máximo nivel de abstracción para la realización única de análisis sintáctico sería la siguiente:

Token	Patrón	Atributos
BLOCK	"block"	
ENDBLOCK	"endblock"	
PYC	";"	
ASIGNACION	id"="{Exp}	
DO	"do"	
UNTIL	"until"	

4. (0.8 puntos) Construir la gramática abstracta para la tabla de tokens obtenida en el caso (a) del ejercicio anterior, eliminar la recursividad y factorizarla, si fuese necesario, y construir la tabla de análisis LL(1) para la gramática transformada.

La gramática abstracta es la siguiente:

$$
\begin{array}{rcl}
\text{Programa} & \longrightarrow & \text{BLOCK Sentencias ENDBLOCK} \\
\text{Sentencias} & \longrightarrow & \text{Sentencias PYC Sentencia} \\
 & | & \text{Sentencia} \\
\text{Sentencia} & \longrightarrow & \text{ID ASIG Exp} \\
 & | & \text{DO Sentencia UNTIL Exp} \\
 & | & \text{Programa} \\
\text{Exp} & \longrightarrow & \text{Exp MASDIV Exp} \\
 & | & \text{Exp MENPOR Exp} \\
 & | & \text{MENPOR Exp} \\
 & | & \text{DIR Exp} \\
 & | & \text{CONST} \\
 & | & \text{ID}
\end{array}
$$

Primero habría que eliminar la recursividad a la izquierda en las producciones que definen `Sentencias` y `Exp`.

Para el caso de `Sentencias` las producciones quedarían así:

$$
\begin{array}{rcl}
\text{Sentencias} & \longrightarrow & \text{Sentencia Sentencias'} \\
\text{Sentencias'} & \longrightarrow & \text{PYC Sentencia Sentencias'} \\
 & | & \epsilon
\end{array}
$$

Para el caso de `Exp` las producciones quedarían así:

$$
\begin{array}{rcl}
\text{Exp} & \longrightarrow & \text{MENPOR Exp Exp'} \\
 & | & \text{DIR Exp Exp'} \\
 & | & \text{CONST Exp'} \\
 & | & \text{ID Exp'} \\
\text{Exp'} & \longrightarrow & \text{MASDIV Exp Exp'} \\
 & | & \text{MENPOR Exp Exp'} \\
 & | & \epsilon
\end{array}
$$

No es necesario factorizar, por lo que la gramática resultante es la siguiente:

1)	Programa	$\longrightarrow$	BLOCK Sentencias ENDBLOCK
2)	Sentencias	$\longrightarrow$	Sentencia Sentencias'
3)	Sentencias'	$\longrightarrow$	PYC Sentencia Sentencias'
4)		$\mid$	ϵ
5)	Sentencia	$\longrightarrow$	ID ASIG Exp
6)		$\mid$	DO Sentencia UNTIL Exp
7)		$\mid$	Programa
8)	Exp	$\longrightarrow$	MENPOR Exp Exp'
9)		$\mid$	DIR Exp Exp'
10)		$\mid$	CONST Exp'
11)		$\mid$	ID Exp'
12)	Exp'	$\longrightarrow$	MASDIV Exp Exp'
13)		$\mid$	MENPOR Exp Exp'
14)		$\mid$	ϵ

La tabla de análisis LL(1) es la siguiente (obtenida con sefalas):

	BLOCK	ENDBLOCK	PYC	ID	ASIG	DO	UNTIL	CONST	MENPOR	MASDIV	DIR	$
Programa	1											
Sentencias	2			2		2						
Sentencia	7			5		6						
Exp				11				10	8		9	
Sentencias'		4	3									
Exp'		14	14				14		13/14	12/14		

Tal y como se muestra en la tabla, aparecen dos celdas con conflictos, por lo tanto, **no es LL(1)**.

5. (0.5 puntos) Dada la gramática con las producciones siguientes:

A	$\longrightarrow$	A b B Z
	$\mid$	ϵ
B	$\longrightarrow$	c d
	$\mid$	B d b
	$\mid$	ϵ
Z	$\longrightarrow$	g Z
	$\mid$	ϵ

Realizar los siguientes cálculos:

(a) (0.2 puntos) Si hacemos análisis LR(1), sea $I_p = \{[A \longrightarrow Ab \cdot BZ, b], \ldots\}$, construir el estado $goto(I_p, B)$ y obtener las acciones que le corresponde en la tabla de análisis.

Solución:

$$goto(I_p, B) = \{[A \longrightarrow AbB \cdot Z, b], [Z \longrightarrow \cdot gZ, b], [Z \longrightarrow \cdot, b]\}$$

Las acciones en dicho estado sería reducir la producción $Z \longrightarrow \epsilon$ ante el símbolo b y desplazar a otros estados ante los símbolos Z y g.

(b) (0.2 puntos) Suponiendo que hacemos análisis SLR y apareciera en un estado I_j el item $[B \longrightarrow cd\cdot]$, obtener las acciones que se deberían introducir en la tabla de análisis.

Solución: Reducir la producción $B \longrightarrow cd$ en $Seguidores(B) = \{g, d, b, \$\}$.

(c) (0.1 puntos) Si hacemos análisis SLR y aparece en el estado I_1 el item $[A' \longrightarrow A\cdot]$, obtener las acciones que se deberáin introducir en la tabla de análisis (siendo A' el símbolo inicial de la gramática aumentada).

Solución: Aceptar la gramática ante el símbolo $.

10. (1.1 puntos) A partir del archivo "host.ugr.es-access_log" de todas las peticiones a través del servidor web, se pretende controlar todos los accesos a cada dirección IP. El formato de dicho archivo se muestra en el examen.

(a) (0.7 puntos) Definir la gramática con atributos para poder controlar qué dirección IP ha solicitado el mayor número de accesos.

Solución: La gramática que define la sintaxis del archivo log es la siguiente:

$$\begin{array}{rcl} \text{Archivo\_log} & \longrightarrow & \text{Archivo\_log linea} \\ & | & \text{linea} \\ \text{linea} & \longrightarrow & \text{IP DATE HOUR NUM REQUEST NAV} \end{array}$$

Dado que se nos piden consideraciones sobre IP y NAV, el resto de los campos podrían considerarse de manera agrupada por simplicidad, sin embargo, se prefiere dejarlo tal y como se define con su significado.

Es necesario almacenar la lista de IPs con sus accesos producidos. Podemos suponer una estructura de datos al estilo de tabla de símbolos (array de registros formados por la pareja de valores IP, accesos). Para ello, se definen las siguientes funciones de uso de dicha estructura de datos:

- `int existeIP(char *IP)`: Busca la dirección IP dada como argumento, la busca en la estructura de datos y devuelve la posición del array, si ha sido encontrada y -1 en caso contrario.
- `void insertaIP(char *IP)`: Crea una nueva entrada para la IP dada como argumento y fija su número de accesos en 1.
- `void incAccesoIP(int indice)`: Accede a la posición del array indicado por el argumento e incrementa el número de accesos en dicha entrada.

El atributo que se evalúa en la dirección IP es de tipo cadena de caracteres. La acción semántica que habría que añadir en la gramática sería la siguiente (suponemos inicializada la estructura de datos totalmente vacía).

$$\begin{array}{rcl} \text{Archivo\_log} & \longrightarrow & \text{Archivo\_log linea} \\ & | & \text{linea} \\ \text{linea} & \longrightarrow & \text{IP DATE HOUR NUM REQUEST NAV} \end{array}$$

```
{
   if (int pos=existeIP($1)==-1)
      insertaIP($1) ;
   else
      incAccesoIP(pos);
}
```

(b) (0.4 puntos) Definir la gramática con atributos para poder controlar el número de accesos realizados por navegadores basados en **Mozilla** y el número de accesos realizados por navegadores basados en **iexplore**.

Solución: Existen varias formas para una correcta solución. La que se propone se basa en dos variables `numIexplore` y `numMozilla` ambos de tipo entero que alojarán el número de accesos registrados en ambos casos. El símbolo terminal NAV es de tipo cadena de caracteres.

Se considera la función `int contiene(char *cad, char *cont)` que devuelve verdadero si la cadena `cont` está contenida en la cadena `cad` y falso en caso contrario.

La gramática con atributos resultante queda como sigue:

$$
\begin{array}{rcl}
\text{Archivo\_log} & \longrightarrow & \text{Archivo\_log linea} \\
 & | & \text{linea} \\
\text{linea} & \longrightarrow & \text{IP DATE HOUR NUM REQUEST NAV}
\end{array}
$$

```
{
   if (contiene("iexplore", $6))
      numIexplore++ ;

   if (contiene("Mozilla", $6))
      numMozilla++ ;
}
```

Si se deseara imprimir el número de accesos, se podría hacer como una producción nueva de mayor abstracción cuya acción semántica fuera imprimir ambos valores.

También sería posible obtener una solución mediante atributos sintetizados, pero la manera más eficiente sería la que se ha mostrado.