PROBLEMAS

6. [1.0 puntos] Dada la siguiente gramática que representa la declaración de tipos registro según la sintaxis de Pascal

registros	→	registros registro
	|	registro
registro	→	**type** nombre = **record** lista_campos **end** pyc
lista_campos	→	lista_campos campo
	|	campo
campo	→	nombre **:** tipo pyc
tipo	→	**integer** | **real** | **character** | **boolean** | nombre
nombre	→	nombre (caracter|digito)
	|	caracter
caracter	→	a|b|c|···|z
digito	→	0|1|2|···|9
pyc	→	**;**

Obtener la tabla de tokens con el máximo nivel de abstracción suponiendo que se va a realizar:

(a) [0.7 puntos] Traducción de un texto en sintaxis de Pascal a otro con sintaxis de C.

Solución:

La tabla de tokens con el máximo nivel de abstracción cuando se realiza el proceso completo de traducción sería la siguiente:

Nº	Token	Patrón
1	TYPE	"type"
2	ASIGNRECORD	"="{sep}*"record"
3	END	"end"
4	PYC	";"
5	DPTOS	":"
6	TIPO	"integer"|"real"|"character"|"boolean"
7	NOMBRE	[a-z]+([a-z]|[0-9])*

donde {sep} es cualquier secuencia de espacios en blanco, tabuladores o retornos de carro.

En cuanto a los atributos, tan sólo el token TIPO es susceptible de llevarlos. En concreto, uno para cada tipo básico. Para contemplar que el tipo de uno de los campos de un registro se basa en un tipo definido por el usuario, no es posible situar el patrón {nombre} como parte del token TIPO porque en otro apartado del lenguaje aparece con significado propio. En este caso, ya que los tipos de un campo pueden ser básicos o definidos, habrá una producción para los básicos y otra para el caso de un nombre definido por el usuario.

(b) [0.3 puntos] Sólo análisis sintáctico.

Solución:

Cuando únicamente se va a realizar análisis sintáctico, primero se observa que la gramática abstracta resultante del proceso anterior es libre de contexto debido a una producción en la forma de construcción de un `registro`:

Nº	Token	Patrón
1	REGISTROS	{registro}*

donde {registro} se define de la siguiente forma usando notación Lex:

```
sep       [ \n\t]*
letra     [a-z]
digito    [0-9]
nombre    {letra}({letra}|{digito})*
tipo      "integer"|"real"|"character"|"boolean"|{nombre}
campo     {nombre}{sep}":"{sep}{tipo}{sep}";"
registro  {sep}"type"{sep}{nombre}{sep}"="    \
          {sep}"record"{sep}{campo}+{sep}"end"{sep}";"
%%
{sep}              ;
{registro}*       ;
.                  printf (''[Linea %d]: Error leyendo caracter %s\n'', yylineno, yytex
%%
```

7. [0.5 puntos] Escriba la gramática abstracta del supuesto (a) del ejercicio anterior, realice los cambios que se estimen oportunos sobre la misma y construya la tabla de análisis LL(1):

Solución:

La gramática abstracta para el supuesto (a) del ejercicio anterior, usando notación YACC es la siguiente:

```
%token TYPE ASIGNRECORD END PYC DPTOS TIPO NOMBRE
%start registros
%%
registros    : registros registro
             | ;
registro     : TYPE NOMBRE ASIGNRECORD lista_campos END PYC ;
lista_campos : lista_campos campo
             | campo ;
campo        : NOMBRE DPTOS tipo_nombre PYC ;
tipo_nombre  : TIPO
             | NOMBRE ;
%%
```

Inicialmente se aprecian varias producciones en las que hay recursividad a la izquierda. En concreto serían las siguientes:

```
registros    : registros registro
             | ;
```

y también en:

```
lista_campos : lista_campos campo
             | campo ;
```

Para la primera es posible realizar un cambio tan simple como intercambiar los símbolos no terminales que intervienen en la correspondiente producción `registros` y `registro`. de esa forma se resuelve el problema de la recursividad a la izquierda. El resultado sería:

```
registros    : registro registros
             | ;
```

Para la segunda es posible realizar lo mismo que en el caso anterior, salvo que ahora se produce el problema de la factorización. Por lo tanto, aplicando la regla de intercambios entre los símbolos no terminales `campo` y `lista_campos` y la regla de sustitución para el problema de la factorización nos quedaría así:

```
lista_campos  : campo lista_campos2 ;
lista_campos2 : lista_campos
              | ;
```

La gramática abstracta resultante final para la realización de la tabla de análisis LL(1) quedaría así:

```
%token TYPE ASIGNRECORD END PYC DPTOS TIPO NOMBRE
%start registros
%%
(1) registros     : registro registros
(2)               | ;
(3) registro      : TYPE NOMBRE ASIGNRECORD lista_campos END PYC ;
(4) lista_campos  : campo lista_campos2 ;
(5) lista_campos2 : lista_campos
(6)               | ;
(7) campo         : NOMBRE DPTOS tipo_nombre PYC ;
(8) tipo_nombre   : TIPO
(9)               | NOMBRE ;
%%
```

La tabla de análisis para análisis sintáctico descendente LL(1) quedaría así:

	TYPE	NOMBRE	ASIGNRECORD	END	PYC	DPTOS	TIPO	$
registros	(1)							(2)
registro	(3)							
lista_campos		(4)						
lista_campos2		(5)		(6)				
campo		(7)						
tipo_nombre		(9)					(8)	

No presenta conflictos la tabla de análisis, por lo tanto, se trata de una gramática LL(1).

8. 0.25 puntos Escriba la gramática abstracta del ejercicio 6(a), construya el estado I_0 para el caso de análisis LR(1) y los estados que son alcanzados por una transición desde ese estado I_0. ¿Habría conflictos en la fila del estado 0 de la tabla de análisis?

Solución:

La gramática abstracta para el ejercicio 6(a), usando notación YACC, es la siguiente:

```
%token TYPE ASIGNRECORD END PYC DPTOS TIPO NOMBRE
%start registros
%%
(1) registros    : registros registro
(2)              | ;
(3) registro     : TYPE NOMBRE ASIGNRECORD lista_campos END PYC ;
(4) lista_campos : lista_campos campo
(5)              | campo ;
(5) campo        : NOMBRE DPTOS tipo_nombre PYC ;
(6) tipo_nombre  : TIPO
(7)              | NOMBRE ;
%%
```

Considerando $[registros' \rightarrow registros]$ como la producción de la gramática aumentada, calculamos el estado I_0 del AFD que modela los prefijos viables para análisis LR(1) de la siguiente forma:

$$\begin{aligned} I_0 = \{ \quad & [registros' \rightarrow \cdot registros,\ \$], \\ & [registros \rightarrow \cdot registros\ registro,\ \$/\texttt{TYPE}], \\ & [registros \rightarrow \cdot,\ \$/\texttt{TYPE}] \ \} \end{aligned}$$

Las transiciones desde el estado I_0 sería tras realizar la reducción del símbolo no terminal `registros`:

$$\begin{aligned} I_1 = goto(I_0, registros) = \{ \quad & [registros' \rightarrow registros\cdot,\ \$], \\ & [registros \rightarrow registros \cdot registro,\ \$/\texttt{TYPE}], \\ & [registro \rightarrow \cdot\texttt{TYPE NOMBRE ASIGNRECORD lista\_campos END PYC},\ \$/\texttt{TYPE}] \ \} \end{aligned}$$

La primera fila de la tabla de análisis LR(1) quedaría así:

Est.	TYPE	ASIGNRECORD	END	PYC	DPTOS	TIPO	NOMBRE	$	registros	registro	lista_campos	campo	tipo_nombre
0	r2							r2	1				

Como puede apreciarse, no existen conflictos en dicha fila, según se solicita en el ejercicio.