EXAMEN DE COMPILADORES (2° **Grado en Informática, final junio-2011**)

FACULTAD DE INFORMÁTICA

Apellidos, nombre: **GRUPO:**

D.N.I.:

Este enunciado y todos los folios usados deben entregarse al salir

Parte I: PREGUNTAS TIPO TEST. 30 %.

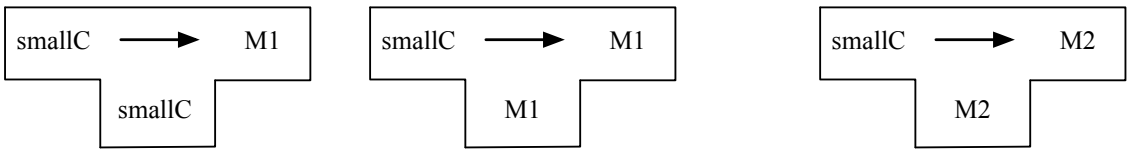
Cada respuesta correcta vale 0.2 puntos.

Cada dos respuestas incorrectas anulan una correcta.

1. c)
2. b)
3. b)
4. b)
5. a)
6. b)
7. c)
8. a)
9. c)
10. b)
11. b)
12. b), c)
13. b)
14. a)
15. a)

Parte II: PREGUNTAS CORTAS. 10 %.

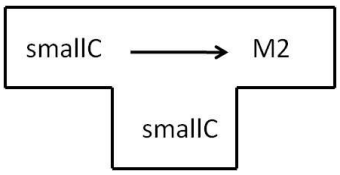
1. *(0.5 puntos)* Se dispone de un compilador del lenguaje fuente *smallC* para la máquina *M1*. Dicho compilador ha sido escrito en el lenguaje de implementación *smallC* (configuración de partida). Se pretende crear una versión del compilador de *smallC* para la máquina *M2* (configuración objetivo). Para hacerlo no se dispone de ningún otro compilador distinto al indicado en la configuración de partida. ¿Cuál es el proceso más adecuado para llegar a la configuración objetivo? Indica las etapas de dicho proceso.



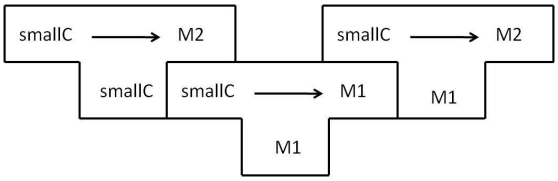
a) Configuración de partida b) Configuración objetivo

Solución:

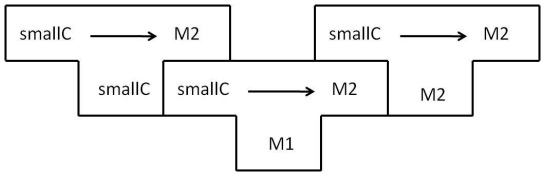
a) Se modifica el generador de código para generar código para la máquina M2:



b) Se compila el compilador modificado, utilizando el compilador original, para obtener un compilador cruzado:



c) Finalmente, se puede utilizar este compilador cruzado para compilarse a sí mismo:



2. *(0.5 puntos)* La siguiente gramática describe sentencias con asignaciones múltiples. ¿Qué modificaciones harías en la gramática para llegar a una equivalente que pudiese ser reconocida con un analizador LL(1)?

$$\begin{array}{lcl} S & \rightarrow & SA \mid A \\ A & \rightarrow & id = L; \\ L & \rightarrow & num \mid id = L \end{array}$$

Solución:

La gramática *no es ambigua*, está *factorizada*, pero es *recursiva por la izquierda*.

Eliminamos, por tanto, la recursividad por la izquierda. Aplicando el algoritmo para los tres símbolos no terminales de la gramática, sólo tenemos que eliminar la recursividad directa de la primera regla de producción (paso 4)a) del algoritmo de los apuntes para i=1) . Obtenemos la siguiente gramática:

$$\begin{array}{lcl} S & \rightarrow & A \mid AS' \\ S' & \rightarrow & A \mid AS' \\ A & \rightarrow & id = L; \\ L & \rightarrow & num \mid id = L \end{array}$$

Esta gramática se puede simplificar de forma obvia, obteniendo:

$$\begin{array}{lcl} S & \rightarrow & A \mid AS \\ A & \rightarrow & id = L; \\ L & \rightarrow & num \mid id = L \end{array}$$

Sin embargo, ahora la gramática no está factorizada. Factorizamos:

$$\begin{array}{lcl} S & \rightarrow & AS' \\ S' & \rightarrow & S \mid \lambda \\ A & \rightarrow & id = L; \\ L & \rightarrow & num \mid id = L \end{array}$$

Y obtenemos una gramática LL(1). esto podría comprobarse con el cálculo de los conjuntos *predict* o la construcción de la tabla.

Parte III: PROBLEMAS. 60 %

Supongamos que se desea formalizar un lenguaje de matrices numéricas en el que, por ejemplo, una matriz de dos filas y tres columnas como

$$\begin{pmatrix} -1 & 3 & 7 \\ 4 & 6 & 0 \end{pmatrix}$$

quedaría representada de la siguiente forma:

$$(\ -1\ 3\ 7\ ;\ 4\ 6\ 0\ ;\ )$$

La siguiente gramática, G, con $V_T = \{(,), num, ;\}$, $V_N = \{MATRIZ, FILA, FILAS, NUMEROS\}$, símbolo inicial $MATRIZ$ y el siguiente conjunto P de producciones:

$$\begin{array}{lll} (1)\ MATRIZ & \rightarrow & (\ FILA\ FILAS\ ) \\ (2)\ FILA & \rightarrow & num\ NUMEROS\ ; \\ (3)\ FILAS & \rightarrow & FILA\ FILAS \\ (4) & | & \lambda \\ (5)\ NUMEROS & \rightarrow & num\ NUMEROS \\ (6) & | & \lambda \end{array}$$

sirve para generar matrices con el formato anterior.
Realiza los siguientes ejercicios:

1. *(1.5 puntos)* Comprueba si se trata de una gramática LL(1) calculando los conjuntos PRIMERO y SIGUIENTE para cada símbolo no terminal, los conjuntos *predict* para cada regla y la tabla de análisis.

   Solución:

   $PRIMERO(M) = \{(\}$ $\quad SIGUIENTE(M) = \{\$\}$
   $PRIMERO(F) = \{num\}$ $\quad SIGUIENTE(F) = \{num,)\}$
   $PRIMERO(FS) = \{num, \lambda\}$ $\quad SIGUIENTE(FS) = \{)\}$
   $PRIMERO(NS) = \{num, \lambda\}$ $\quad SIGUIENTE(NS) = \{;\}$
   $predict(1) = \{(\}$ $\quad predict(4) = \{)\}$
   $predict(2) = \{num\}$ $\quad predict(5) = \{num\}$
   $predict(3) = \{num\}$ $\quad predict(6) = \{;\}$

NO TERM	**TERMINAL**				
	(	)	**num**	;	$
M	1				
F			2		
FS		4	3		
NS			5	6	

   La gramática es LL(1), pues no aparecen conflictos en la tabla.

2. *(0.5 puntos)* Simular el comportamiento del algoritmo de análisis LL para la cadena w ≡ (num num ; ;), realizando la recuperación de errores en modo pánico en caso de error.

   Solución:

PILA	**ENTRADA**	**SALIDA**
$\$M$	$(num\ num\ ;\ ;\ )\$$	1
$\$)FS\ F($	$(num\ num\ ;\ ;\ )\$$	
$\$)FS\ F$	$num\ num\ ;\ ;\ )\$$	2
$\$)FS;NS\ num$	$num\ num\ ;\ ;\ )\$$	
$\$)FS;NS$	$num\ ;\ ;\ )\$$	5
$\$)FS;NS\ num$	$num\ ;\ ;\ )\$$	
$\$)FS;NS$	$;\ ;\ )\$$	6
$\$)FS;$	$;\ ;\ )\$$	
$\$)FS$	$;\ )\$$	$ERROR\ ; \notin PRIMERO(FS)\ y\ ; \notin SIGUIENTE(FS).$ $)\notin PRIMERO(FS)\ y\ ) \in SIGUIENTE(FS).$ *Extraemos ; de la entrada y sacamos FS de la pila.*
$\$)$	$)\$$	
$\$$	$\$$	*Acepta*

3. *(1.75 puntos)* Construir la colección LR(1) para G y la tabla LR-canónica. Indicar si G es una gramática LR-canónica justificando la respuesta.

Solución:

Aumentamos la gramática con el nuevo símbolo inicial M' y la regla $M' \to M$.

$I_0 = \{[M' \to \cdot M, \$], [M \to \cdot(F\ FS), \$]\}$

$GOTO(I_0, M) = I_1 = \{[M' \to M\cdot, \$]\}$

$GOTO(I_0, () = I_2 = \{[M \to (\cdot F\ FS), \$], [F \to \cdot num\ NS;, num/)]\}$

$GOTO(I_2, F) = I_3 = \{[M \to (F \cdot FS), \$], [FS \to \cdot F\ FS,)], [FS \to \cdot,)], [F \to \cdot num\ NS;, num/)]\}$

$GOTO(I_2, num) = I_4 = \{[F \to num \cdot NS;, num/)], [NS \to \cdot num\ NS, ;], [NS \to \cdot, ;]\}$

$GOTO(I_3, FS) = I_5 = \{[M \to (F\ FS\cdot), \$]\}$

$GOTO(I_3, F) = I_6 = \{[FS \to F \cdot FS,)], [FS \to \cdot F\ FS,)], [FS \to \cdot,)], [F \to \cdot num\ NS;, num/)]\}$

$GOTO(I_3, num) = I_4$

$GOTO(I_4, NS) = I_7 = \{[F \to num\ NS\cdot;, num/)\}$

$GOTO(I_4, num) = I_8 = \{[NS \to num \cdot NS, ;], [NS \to \cdot num\ NS, ;], [NS \to \cdot, ;]\}$

$GOTO(I_5,)) = I_9 = \{[M \to (F\ FS\ )\cdot, \$]\}$

$GOTO(I_6, FS) = I_{10} = \{[FS \to F\ FS\cdot,)]\}$

$GOTO(I_6, F) = I_6$

$GOTO(I_6, num) = I_4$

$GOTO(I_7, ;) = I_{11} = \{[F \to num\ NS;\cdot, num/)\}$

$GOTO(I_8, NS) = I_{12} = \{[NS \to num\ NS\cdot, ;]\}$

$GOTO(I_8, num) = I_8$

ESTADO	**accion**					**ir_a**			
	(	)	**num**	;	$	**M**	**F**	**FS**	**NS**
0	$d2$					1			
1					*aceptar*				
2			$d4$				3		
3		$r4$	$d4$				6	5	
4			$d8$	$r6$					7
5		$d9$							
6		$r4$	$d4$				6	10	
7				$d11$					
8			$d8$	$r6$					12
9					$r1$				
10		$r3$							
11		$r2$	$r2$						
12				$r5$					

Puesto que no aparecen conflictos en la tabla, la gramática es LR-Canónica.

4. *(0.75 puntos)* Indicar si G es una gramática LALR y/o SLR justificando la respuesta, y sin calcular ninguna colección de items adicional.

No hay estados que se puedan unir para obtener un nuevo autómata LALR. Por tanto, la tabla anterior es también la tabla LALR y, puesto que no hay conflictos, la gramática es LALR.

Por otro lado, la tabla SLR sólo difiere de la LALR en que las reducciones para cada regla $A \to \alpha$ se realizan cuando en la entrada aparece un token que pertenezca a $SIGUIENTE(A)$, mientras que en la LALR se reduce para un subconjunto (no propio) de esos conjuntos de tokens, que aparecen como segundo componente en cada item. En este caso, si observamos la tabla, vemos que:

- La reducción con la regla (1) $MATRIZ \to (\ FILA\ FILAS\ )$ se realiza para el símbolo \$ (en el estado 9) y $SIGUIENTE(MATRIZ) = \{\$\}$.
- Las reducciones con la regla (2) $FILA \to num\ NUMEROS\ ;$ se realizan para los símbolos $\{), num\}$ (en el estado 11) y
  $SIGUIENTE(FILA) = \{num,)\}$.
- La reducción con la regla (3) $FILAS \to FILA\ FILAS$ se realiza para el símbolo) (en el estado 10) y $SIGUIENTE(FILAS) = \{)\}$.

- Las reducciones con la regla (4) $FILAS \rightarrow \lambda$ se realizan para el símbolo) (en los estados 3 y 6) y $SIGUIENTE(FILAS) = \{)\}$.
- La reducción con la regla (5) $NUMEROS \rightarrow num\ NUMEROS$ se realiza para el símbolo ; (en el estado 12) y
  $SIGUIENTE(NUMEROS) = \{;\}$.
- Las reducciones con la regla (6) $NUMEROS \rightarrow \lambda$ se realizan para el símbolo ; (en los estados 4 y 8) y $SIGUIENTE(NUMEROS) = \{;\}$.

Es decir, coinciden exactamente los conjuntos de tokens para los que se reduce en la tabla LALR con aquellos con los que se reduce en la tabla SLR. Por tanto, ambas tablas son idénticas y la gramática es también SLR.

5. *(1.5 puntos)* Dar una *definición dirigida por la sintaxis* para añadir la restricción de contexto que permita controlar que todas las filas de la matriz tengan el mismo número de elementos, de manera que se llame a una función *error* en el caso de que alguna fila no tenga el mismo número de elementos que la primera. Para esto:

   - indicar el número y tipo de atributos asociado a cada símbolo de G.
   - asociar a cada regla de producción de G las acciones semánticas necesarias.
   - decorar el árbol sintáctico correspondiente a la cadena $w \equiv (\ -1\ 3\ 7\ ;\ 4\ 6\ 0\ ;\ )$.
   - indicar si G es S-atribuida y/o L-atribuida justificando la respuesta.

   Solución:

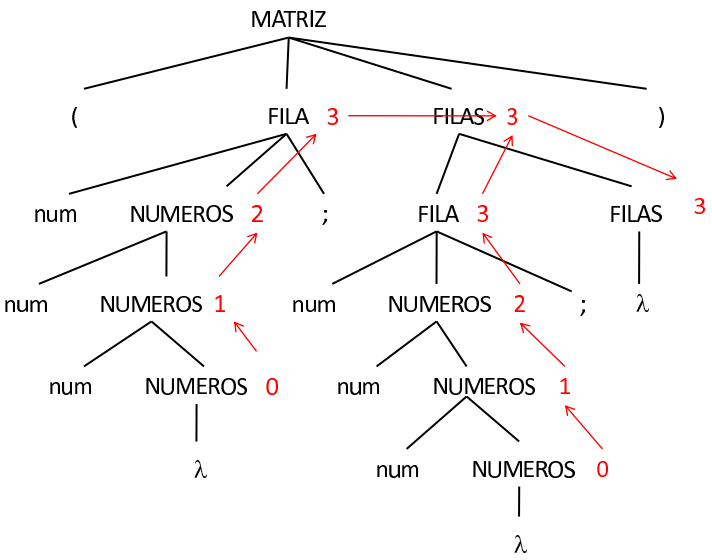
   Consideramos los siguientes atributos:

   FILA → Columnas (*sintetizado*).
   FILAS → PrimeraColumnas (*heredado*).
   NUMEROS → Num (*sintetizado*).

   La *definición dirigida por la sintaxis* podría ser la siguiente:

(1)	$MATRIZ$	→	$(\ FILA\ FILAS\ )$ $\{FILAS.PrimeraColumnas = FILA.Columnas;\}$
(2)	$FILA$	→	$num\ NUMEROS\ ;$ $\{FILA.Columnas = NUMEROS.Num + 1;\}$
(3)	$FILAS$	→	$FILA\ FILAS_1$ $\{if(FILAS.PrimeraColumnas == FILA.Columnas)$ $FILAS_1.PrimeraColumnas = FILAS.PrimeraColumnas$ $else\ error()\}$
(4)		|	λ
(5)	$NUMEROS$	→	$num\ NUMEROS_1$ $\{NUMEROS.Num = NUMEROS_1.Num + 1;\}$
(6)		|	λ $\{NUMEROS.Num = 0;\}$

   El árbol decorado para la sentencia $w \equiv (\ -1\ 3\ 7\ ;\ 4\ 6\ 0\ ;\ )$ sería:



La gramática no es S-Atribuida, puesto que tiene atributos heredados. Sí es L-Atribuida, pues todos los atributos son o *sintetizados* o *heredados* del padre o del hermano por la izquierda.