

# Lenguajes de Programación y Procesadores de Lenguajes

Ejercicios      curso 2021 - 2022

**José Miguel Benedí**  
<jbenedi@dsic.upv.es>

7 de noviembre de 2021

# Índice

<b>2. Análisis Léxico</b>	<b>1</b>
2.1. Ejercicios resueltos . . . . .	2
<b>3. Análisis Sintáctico</b>	<b>4</b>
3.1. Introducción . . . . .	4
3.1.1. Ejercicios resueltos . . . . .	5
3.2. Análisis Sintáctico Descendente . . . . .	7
3.2.1. Ejercicios resueltos . . . . .	10
3.3. Análisis Sintáctico Ascendente . . . . .	17
3.3.1. Ejercicios resueltos . . . . .	20
<b>4. Análisis Semántico</b>	<b>24</b>
4.1. Gramáticas de Atributos . . . . .	24
4.1.1. Ejercicios resueltos . . . . .	26
4.2. Comprobación de Tipos . . . . .	29
4.2.1. Ejercicios resueltos . . . . .	30
<b>5. Gestión de Memoria</b>	<b>32</b>
5.1. Ejercicios resueltos . . . . .	36
<b>6. Generación de Código Intermedio</b>	<b>40</b>
6.1. Ejercicios resueltos . . . . .	43
<b>7. Optimización de Código Intermedio</b>	<b>45</b>
7.1. Ejercicios resueltos . . . . .	49

## 2. Análisis Léxico

1. Dado el alfabeto de los dígitos proporcionad una expresión regular que defina el lenguaje de todas las cadenas de dígitos que empiecen y terminen por 2.
2. \* Dado el alfabeto de los dígitos proporcionad una expresión regular que defina el lenguaje de todas las cadenas de dígitos que representen números pares.
3. \* Dado el alfabeto  $\{a, b, c\}$ , proporcionad una expresión regular que defina el lenguaje de todas las cadenas que contengan exactamente una  $a$ .
4. \* Dado el alfabeto  $\{a, b, c\}$ , proporcionad una expresión regular que defina el lenguaje de todas las cadenas que no contengan dos  $a$  consecutivas.
5. \* Dado el alfabeto  $\{a, b, c\}$ , proporcionad una expresión regular en la que la primera aparición de  $b$  es siempre precedida por al menos una aparición de  $a$ .
6. \* Dado el alfabeto  $\{a, b, c\}$ , proporcionad una descripción concisa en Castellano del lenguaje que genera la expresión regular:  $((a|b)^* c (a|b)^* c)^* (a|b)^*$ .
7. \* Proporcionar una expresión regular en la que todas las cadenas de letras minúsculas deben contener las cinco vocales en orden (cada vocal solo debe aparecer una vez); por ejemplo: `k j a h l e i b o k j u m`
8. Dibujad un AEF para las expresiones regulares de los ejercicios 2 a 8.
9. \* Dada la siguiente cadena de entrada `babcaababccbcabb` vuestro AL detecta los siguientes tokens: `b abc aab abcc b c abb`  
Proporcionad una especificación léxica en términos de expresiones regulares. Emplead solo las operaciones de concatenación y cierre transitivo y reflexivo.
10. Diseñad una expresión regular para las siguientes especificaciones en lenguaje natural:
  - a) La dirección de un correo electrónico.
  - b) La dirección de una página web.
11. \* Describid brevemente las principales funciones de un Analizador Léxico e indicad cómo se puede tratar el problema de la detección de las palabras reservadas.
12. \* Construid una *expresión regular* (o en formato FLEX, si se quiere) para las constantes numéricas, que permita, por ejemplo: `3.14 3. 3 0.14`
13. \* Construid una *expresión regular* en formato FLEX para los números sin signo del lenguaje PASCAL.
14. Escribid un programa FLEX que sustituya las palabras reservadas `while`, `do`, `if`, `else` a mayúsculas. Debe generar un fichero igual al de la entrada con la única diferencia del cambio de dichas palabras reservadas.
15. Construid una expresión regular para las constantes numéricas decimales; por ejemplo: `2.27 -4e7 +14.0e-4 .127e+7`

16. Diseñad las expresiones regulares para las constantes numéricas de tu lenguaje de programación favorito.
17. Construid un programa FLEX para la especificación léxica de tu lenguaje de programación favorito.
18. Construid un programa FLEX para la especificación léxica del lenguaje **MenosC.18** (consultad el enunciado del proyecto).

## 2.1. Ejercicios resueltos

**Ejercicio 2**      $digito$       $[0 - 9]$   
                    $pares$       $[02468]$   
                    $\{digito\} * \{pares\}$

**Ejercicio 3**      $[bc] * a [bc]^*$

**Ejercicio 4**      $[bc] * ((a [bc]^+)^* \mid (a [bc]^*)^*)$

**Ejercicio 5**      $[ac] * ab [abc]^* \mid [ac]^*$

**Ejercicio 6**     Cadenas con un número par de  $c$ .

**Ejercicio 7**      $noV$       $[\wedge aeiou]$   
                    $\{noV\} * a \{noV\} * e \{noV\} * i \{noV\} * o \{noV\} * u \{noV\}^*$

**Ejercicio 9**     Dos posibles soluciones correctas

$$\begin{array}{l|l} 1 & \begin{array}{l} c \\ b \\ aa^*b^*c^* \end{array} \\ 2 & \begin{array}{l} aa^*b^*c^* \\ c^*b^* \end{array} \end{array}$$

**Ejercicio 11**     Las principales funciones de un Analizador Léxico (AL) son la detección de los símbolos del lenguaje y la realización de las acciones semánticas asociadas con la detección de un símbolo, así como la emisión de las componentes léxicas (“tokens”) detectadas.

Además el AL también realiza otras funciones: 1) Tratamiento de errores léxicos; 2) eliminación de cadenas inútiles (comentarios, tabuladores, saltos de línea, etc.); 3) lectura eficiente del fichero de entrada; 4) relación de los mensajes de error con las líneas del programa fuente; y 4) reconocimiento y ejecución de las directivas de compilación.

Las palabras reservadas pueden tratarse, o bien con expresiones regulares independientes, o bien, como identificadores especiales (tabla de palabras reservadas).

**Ejercicio 12**      $digito$       $[0 - 9]$   
                    $\{digito\} + ( "." \{digito\}^*)^?$

**Ejercicio 13**

<i>digito</i>	$[0 - 9]$
<i>digitos</i>	$\{digitos\}^+$
<i>fracOpc</i>	$(\text{"."}\{digitos\})^?$
<i>expoOpc</i>	$(\text{"E"}([+-])^?\{digitos\})^?$
<i>numeros</i>	$\{digitos\}\{fracOpc\}\{expoOpc\}$

### 3. Análisis Sintáctico

#### 3.1. Introducción

19. Dado el alfabeto  $\Sigma = \{a, b\}$ , proporcionad una expresión regular y una gramática incontextual para el lenguaje formado por todas las cadenas que empiecen por  $a$ .
20. Proporcionad un ejemplo simple de lenguaje que sea incontextual y no sea regular.
21. \* Dada la gramática que se muestra, calculad una secuencia de derivación a izquierdas para cada una de las siguientes tres cadenas:  $\{\{i\}, \{i, i\}\} \quad \{i, i, i\} \quad \{i, \{i\}\}$

$$B \Rightarrow \{I\} \mid i \qquad I \Rightarrow I, B \mid B$$

22. \* Para la misma gramática del ejercicio 21, calculad una secuencia de derivación a derechas para cada una de las mismas tres cadenas.
23. Construid el árbol de análisis sintáctico asociado a la primera cadena de los ejercicios 21 y 22. ¿Qué se puede decir de ambos y porqué?
24. Dada la siguiente gramática, calculad la secuencia de derivación a izquierdas y a secuencia de derivación a derechas para la cadena:  $a a + a *$

$$S \Rightarrow S S + \mid S S * \mid a$$

Obtened un árbol de análisis para la cadena. ¿Es la gramática ambigua? justificad la respuesta.

25. Dada la siguiente gramática, calculad la secuencia de derivación a izquierdas y la secuencia de derivación a derechas para la cadena:  $b c d c$

$$S \Rightarrow S B \mid b \qquad B \Rightarrow B d \mid c$$

26. [A.Aho, R.Sethi, J.Ullman, 1990] Dada la gramática que se muestra, demostrad que esta gramática es ambigua construyendo dos árboles de análisis sintáctico distintos para la cadena:  $abab$

$$S \Rightarrow a S b S \mid b S a S \mid \epsilon$$

27. \* Escribid una gramática no ambigua que genere el lenguaje de los palíndromos sobre el alfabeto  $\{a, b\}$ . Considerando esta gramática, obtened la secuencia de derivaciones a izquierdas para la cadena:  $a b b a$
28. \* Escribid una gramática no ambigua que genere el lenguaje de las expresiones booleanas que incluyan: operadores **and**, **or** y **not**; constantes **true** y **false**; y paréntesis ( y ). Considerad la precedencia: **or** menos precedencia que **and** y **and** menos que **not**. Considerando esta gramática, obtened la secuencia de derivaciones a derechas para la cadena:  $not (true or false)$
29. \* Dado el alfabeto  $\Sigma = \{a, b\}$ , escribid una gramática incontextual para cada uno de los siguientes lenguajes:

a) Cadenas del lenguaje  $L: \{a^{2n}b^m c^{3n} \mid n, m \geq 0\}$

b) Conjunto de todas las cadenas con más  $a$  que  $b$ .

30. Escribid una gramática para definir los números romanos menores que 100.

31. \* Dado el autómata a pila que se adjunta, obtened la traza asociada (con el criterio: pila vacía) para la cadena  $a + a * a$  tal que:  $(q, a + a * a, E) \stackrel{*}{\vdash} (q, , )$

$$A_p = (Q, T, \Gamma, \delta, q, E, \emptyset) \quad \text{con} \quad Q = \{q\}; \quad T = \{a, +, *\}; \quad \Gamma = \{E, T, F, a, +, *\}$$

$$\delta(q, a, a) = (q, \epsilon); \quad \delta(q, +, +) = (q, \epsilon); \quad \delta(q, *, *) = (q, \epsilon);$$

$$\delta(q, \epsilon, E) = \{(q, E + T), (q, T)\}; \quad \delta(q, \epsilon, T) = \{(q, T * F), (q, F)\}; \quad \delta(q, \epsilon, F) = (q, a);$$

32. \* Dado el autómata a pila que se adjunta, obtened la traza asociada (con el criterio: pila vacía) para la cadena  $a + a * a$  tal que:  $(q, a + a * a, \$) \stackrel{*}{\vdash} (q, , )$

$$A_p = (Q, T, \Gamma, \delta, q, \$, \emptyset) \quad \text{con} \quad Q = \{q\}; \quad T = \{a, +, *\}; \quad \Gamma = \{E, T, F, a, +, *\} \cup \{\$\}$$

$$\delta(q, a, \epsilon) = (q, a); \quad \delta(q, +, \epsilon) = (q, +); \quad \delta(q, *, \epsilon) = (q, *);$$

$$\delta(q, \epsilon, \$E) = (q, \epsilon); \quad \delta(q, \epsilon, E + T) = (q, E); \quad \delta(q, \epsilon, T) = (q, E);$$

$$\delta(q, \epsilon, T * F) = (q, T); \quad \delta(q, \epsilon, F) = (q, T); \quad \delta(q, \epsilon, a) = (q, F);$$

33. Comparar el ejercicio 21 y el 31 y relacionarlo con el Análisis Sintáctico Descendente.

34. Análogamente, comparar el ejercicio 22 y el 32 y relacionarlo con el Análisis Sintáctico Ascendente.

### 3.1.1. Ejercicios resueltos

**Ejercicio 21** Solo para la primera cadena:

$$B \Rightarrow \{I\} \Rightarrow \{I, B\} \Rightarrow \{B, B\} \Rightarrow \{\{I\}, B\} \Rightarrow \{\{B\}, B\} \Rightarrow \{\{i\}, B\} \Rightarrow \{\{i\}, \{I\}\}$$

$$\Rightarrow \{\{i\}, \{I, B\}\} \Rightarrow \{\{i\}, \{B, B\}\} \Rightarrow \{\{i\}, \{i, B\}\} \Rightarrow \{\{i\}, \{i, i\}\}$$

**Ejercicio 22** Solo para la primera cadena:

$$B \Rightarrow \{I\} \Rightarrow \{I, B\} \Rightarrow \{I, \{I\}\} \Rightarrow \{I, \{I, B\}\} \Rightarrow \{I, \{I, i\}\} \Rightarrow \{I, \{B, i\}\} \Rightarrow \{I, \{i, i\}\}$$

$$\Rightarrow \{B, \{i, i\}\} \Rightarrow \{\{I\}, \{i, i\}\} \Rightarrow \{\{B\}, \{i, i\}\} \Rightarrow \{\{i\}, \{i, i\}\}$$

**Ejercicio 27** Una posible gramática:

$$S \Rightarrow a S a \mid b S b \mid \epsilon$$

Y la secuencia de derivación a izquierdas es:

$$S \Rightarrow a S a \Rightarrow a b S b a \Rightarrow a b b a$$

**Ejercicio 28** Una posible gramática:

$$E \Rightarrow E \text{ or } T \mid T$$

$$T \Rightarrow T \text{ and } F \mid F$$

$$F \Rightarrow \text{not } I \mid I$$

$$I \Rightarrow \text{true} \mid \text{false} \mid ( E )$$

Y la secuencia de derivación a derechas es:

$$\begin{aligned}
 E &\Rightarrow T \Rightarrow F \Rightarrow \text{not } I \Rightarrow \text{not } (E) \Rightarrow \text{not } (E \text{ or } T) \Rightarrow \text{not } (E \text{ or } F) \\
 &\Rightarrow \text{not } (E \text{ or } I) \Rightarrow \text{not } (E \text{ or } \text{false}) \Rightarrow \text{not } (T \text{ or } \text{false}) \\
 &\Rightarrow \text{not } (F \text{ or } \text{false}) \Rightarrow \text{not } (I \text{ or } \text{false}) \Rightarrow \text{not } (\text{true} \text{ or } \text{false})
 \end{aligned}$$

### Ejercicio 29

1. Cadenas del lenguaje  $L: \{a^{2n}b^m c^{3n} \mid n, m \geq 0\}$

$$S \Rightarrow a a S c c c \mid B \qquad B \Rightarrow b B \mid \epsilon$$

2. Conjunto de todas las cadenas con más  $a$  que  $b$ .

$$\begin{aligned}
 S &\Rightarrow A a \mid M S \mid S M A \\
 A &\Rightarrow A a \mid \epsilon \\
 M &\Rightarrow M M \mid b M a \mid a M b \mid \epsilon
 \end{aligned}$$

### Ejercicio 31

$$\begin{aligned}
 (q, a+a*a, E) &\vdash (q, a+a*a, E+T) \vdash (q, a+a*a, T+T) \vdash (q, a+a*a, F+T) \vdash \\
 (q, a+a*a, a+T) &\vdash (q, +a*a, +T) \vdash (q, a*a, T) \vdash (q, a*a, T*F) \vdash \\
 (q, a*a, F*F) &\vdash (q, a*a, a*F) \vdash (q, *a, *F) \vdash (q, a, F) \vdash (q, a, a) \vdash (q, , )
 \end{aligned}$$

### Ejercicio 32

$$\begin{aligned}
 (q, a+a*a, \$) &\vdash (q, +a*a, \$a) \vdash (q, +a*a, \$F) \vdash (q, +a*a, \$T) \vdash \\
 (q, +a*a, \$E) &\vdash (q, a*a, \$E+) \vdash (q, *a, \$E+a) \vdash (q, *a, \$E+F) \vdash \\
 (q, *a, \$E+T) &\vdash (q, a, \$E+T*) \vdash (q, , \$E+T*a) \vdash (q, , \$E+T*F) \vdash \\
 (q, , \$E+T) &\vdash (q, , \$E) \vdash (q, , )
 \end{aligned}$$



### 3.2. Análisis Sintáctico Descendente

35. \* Dada una gramática  $\mathcal{G}$  bien definida (todos los no-terminales son accesibles y no hay símbolos inútiles), demostrad que:  $\forall A \in N, \text{SIGUIENTE}(A) \neq \emptyset$ .
36. \* Calculad PRIMEROS y los SIGUIENTES de todos los símbolos no terminales de la gramática. Comprobar igualmente si cumple la condición LL(1).

$$\begin{array}{lll} S \Rightarrow a A B C & B \Rightarrow a \mid \epsilon & D \Rightarrow c \mid \epsilon \\ A \Rightarrow b b D \mid a & C \Rightarrow b \mid \epsilon & \end{array}$$

37. \* Ídem para la siguiente gramática:

$$\begin{array}{lll} S \Rightarrow E D C & B \Rightarrow \epsilon & D \Rightarrow B \mid b b \\ A \Rightarrow a \mid \epsilon & C \Rightarrow c \mid \epsilon & E \Rightarrow A e \mid \epsilon \end{array}$$

38. \* Ídem para la siguiente gramática:

$$\begin{array}{ll} S \Rightarrow B C D & B \Rightarrow C D \mid D C \\ C \Rightarrow c C \mid \epsilon & D \Rightarrow D d \mid a \mid \epsilon \end{array}$$

39. Calculad los PRIMEROS y los SIGUIENTES de todos los símbolos no terminales de la gramática:

$$\begin{array}{ll} S \Rightarrow a S A \mid \epsilon & A \Rightarrow B b \\ B \Rightarrow A c \mid \epsilon & \end{array}$$

40. Dada una gramática LL(1), demostrad las siguientes proposiciones:

- a) \* Si una gramática es recursiva a izquierdas, entonces no es LL(1).
- b) \* Si una gramática es LL(1), entonces no es ambigua.
41. \* Dada una *derivación a izquierdas*  $S \xRightarrow{*} w A \alpha$  y siendo  $a$  el símbolo actual de la cadena de entrada. ¿Qué parte de la forma sentencial  $w A \alpha$  puede estar en la pila de un analizador LL(1) y qué condición se debe cumplir para que la siguiente acción sea derivar  $A \rightarrow \beta$ ?
42. \* Dada la siguiente gramática: a) Construid la tabla de análisis LL(1), y b) Proporcionad la traza LL(1) para la cadena:  $x z x$

$$\begin{array}{lll} A \rightarrow B C & C \rightarrow z D & E \rightarrow y \mid \epsilon \\ B \rightarrow w D \mid E D & D \rightarrow x D \mid \epsilon & \end{array}$$

43. \* Dada la siguiente gramática, obtened una gramática LL(1) equivalente y comprobad que la nueva gramática es LL(1).

$$P \rightarrow P x \mid y P y \mid y Q \quad Q \rightarrow x \mid z$$

44. \* Para la gramática del ejercicio 21, comprobad si cumple la condición LL(1); en caso negativo, transformad la gramática y construid su tabla de análisis LL(1) ¿Es una gramática LL(1)? En caso afirmativo, proporcionad la traza de análisis para la cadena:  $\{\{i\}, \{i, i\}\}$

45. Ídem para la gramática del ejercicio 37 y la siguiente cadena:  $aabbcb$

46. \* Ídem para la siguiente gramática y las cadenas:  $aab \quad d$

$$\begin{array}{lcl} A & \Rightarrow & B b \mid C d \\ C & \Rightarrow & c C \mid \epsilon \end{array} \qquad \begin{array}{lcl} B & \Rightarrow & a B \mid \epsilon \end{array}$$

47. Ídem para la siguiente gramática y las dos cadenas siguientes:  $c \quad \epsilon$

$$\begin{array}{lcl} A & \Rightarrow & B C \\ C & \Rightarrow & c C \mid \epsilon \end{array} \qquad \begin{array}{lcl} B & \Rightarrow & b c B \mid \epsilon \end{array}$$

48. Ídem para la siguiente gramática y la cadena:  $ca$

$$\begin{array}{lcl} S & \Rightarrow & B \\ C & \Rightarrow & c C \mid \epsilon \end{array} \qquad \begin{array}{lcl} B & \Rightarrow & C D \\ D & \Rightarrow & a \end{array}$$

49. Ídem para la siguiente gramática y la cadena:  $0110$

$$S \Rightarrow 0 S 0 \mid 1 S 1 \mid \epsilon$$

50. \* Ídem para la siguiente gramática y la cadena:  $aabb$

$$\begin{array}{lcl} S & \Rightarrow & A B \\ A & \Rightarrow & a A \mid \epsilon \end{array} \qquad \begin{array}{lcl} B & \Rightarrow & b B \mid \epsilon \end{array}$$

51. Demostrad que las siguientes gramáticas no son LL(1).

$$\begin{array}{lcl} S & \Rightarrow & A B \mid C D x \\ A & \Rightarrow & x y \mid m \\ B & \Rightarrow & z \\ C & \Rightarrow & p C \mid \epsilon \\ D & \Rightarrow & q D \mid \epsilon \end{array} \qquad \begin{array}{lcl} S & \Rightarrow & A b c \mid a A c b \\ A & \Rightarrow & b \mid c \mid \epsilon \\ S & \Rightarrow & a A a \mid \epsilon \\ A & \Rightarrow & a b S \mid C \\ C & \Rightarrow & b C \mid \epsilon \end{array}$$

52. Demostrad que una gramática tiene una tabla de análisis LL(1), sin entradas múltiples, si y solo si la gramática cumple la condición LL(1).

53. Indicad si las siguientes afirmaciones son ciertas o falsas, justificando brevemente la respuesta:

- En una tabla de análisis LL(1) todas las filas (asociadas a los no-terminales) tienen al menos una acción **derivar**.
- En una tabla de análisis LL(1) todas las columnas (asociadas a los terminales) tienen al menos una acción **derivar**.

54. Dada la siguiente gramática, eliminad la recursividad a izquierdas. ¿La gramática resultante cumple la condición LL(1)?

$$S \Rightarrow A b \mid B c \quad A \Rightarrow A a \mid \epsilon \quad B \Rightarrow B a \mid \epsilon$$

55. \* Considerad la siguiente gramática en la que le faltan dos reglas

$$\begin{array}{lll} S \Rightarrow a S \mid \dots(1) & X \Rightarrow c S \mid \epsilon & Z \Rightarrow e S \\ A \Rightarrow \dots(2) \mid \epsilon & Y \Rightarrow d S \mid \epsilon & \end{array}$$

Afortunadamente disponemos del conjunto de PRIMEROS y SIGUIENTES:

	PRIMEROS	SIGUIENTES
S	$\{a, b, c, d, e\}$	$\{\$ \} \cup \text{SIGUIENTES}(X) \cup \text{SIGUIENTES}(Y) \cup \text{SIGUIENTES}(Z)$
A	$\{c, d, e, \epsilon\}$	$\{b\}$
X	$\{c, \epsilon\}$	$(\text{PRIMEROS}(Y) - \epsilon) \cup \text{PRIMEROS}(Z)$
Y	$\{d, \epsilon\}$	$\text{PRIMEROS}(Z)$
Z	$\{e\}$	$\text{SIGUIENTES}(A)$

Reconstruir la gramática completando las reglas que faltan.

56. Comprobad si las siguientes gramáticas son LL(1), en caso contrario tratad de obtener una LL(1) equivalente.

$$\begin{array}{lll} A \Rightarrow a A F \mid a B F \mid a & S \Rightarrow A a \mid b & E \Rightarrow E + E \\ B \Rightarrow B b \mid c & A \Rightarrow S B & E \Rightarrow E * E \\ F \Rightarrow f & B \Rightarrow a b & E \Rightarrow ( E ) \mid a \end{array}$$

57. Demostrad que la transformación que elimina la recursividad a izquierdas produce una gramática equivalente.

$$A \rightarrow A \alpha \mid \beta \quad \Rightarrow \quad A \rightarrow \beta A' \quad A' \rightarrow \alpha A' \mid \epsilon$$

58. Demostrad que la transformación que elimina posible factores comunes produce una gramática equivalente.

$$A \rightarrow \alpha \beta_1 \mid \alpha \beta_2 \mid \gamma \quad \Rightarrow \quad A \rightarrow \alpha A' \mid \gamma \quad A' \rightarrow \beta_1 \mid \beta_2$$

59. \* Demostrar que la siguiente gramática es LL(1), a pesar de que es factorizable a izquierdas:

$$S \Rightarrow A x \mid A \quad A \Rightarrow \epsilon$$

60. \* Dada la siguiente gramática: a) calculad los PRIMEROS y SIGUIENTES de todos los no-terminales; b) construid la tabla de análisis LL(1); y c) ¿es LL(1) y porqué?.

$$\begin{array}{ll} S \Rightarrow U V W & V \Rightarrow w \mid x U \mid \epsilon \\ U \Rightarrow u \mid W v \mid \epsilon & W \Rightarrow y \mid z \end{array}$$

61. \* Dada la siguiente gramática:

$$\begin{array}{ll} S \rightarrow b A b \mid b B a & B \rightarrow b \mid B C \\ A \rightarrow a S \mid C B & C \rightarrow c \end{array}$$

- Sin necesidad de hacer ningún cálculo, dad un par de razones por las que esta gramática no es LL(1)
- Utilizando las transformaciones vistas en clase, reescribid la gramática eliminando los posibles problemas detectados en el apartado anterior.
- Para la gramática resultante, calculad los *primeros* de todas las partes derechas de las reglas y los *siguientes* de todos sus no-terminales.
- Construid la tabla de análisis LL(1). ¿Es una gramática LL(1) y porqué?
- A partir de la tabla LL(1), proporcionad la traza de análisis LL(1) para la cadena:  $\begin{array}{c} b \ b \\ c \ a \end{array}$
- Para una gramática (bien formada), en su tabla de análisis LL(1), ¿es posible que exista una fila (asociada a un no-terminal) que no tenga al menos una acción *derivar*? Justificad la respuesta.

### 3.2.1. Ejercicios resueltos

#### Ejercicio 35

Lo vamos a demostrar por reducción al absurdo: Sea una gramática  $\mathcal{G}$  bien definida y  $\exists A \in N : \text{SIGUIENTE}(A) = \emptyset$ .

En primer lugar,  $A \neq S$  ya que  $\text{SIGUIENTE}(S) = \{\$ \}$ .

Y en segundo lugar, si  $\text{SIGUIENTE}(A) = \emptyset$  esto implica que  $\nexists (B \rightarrow \alpha A \beta) \in \mathcal{P}$  en la que  $A$  se parte derecha de una regla. Si esto ocurre,  $A$  no sería accesible y por tanto  $\mathcal{G}$  no estaría bien definida, lo que contradice la suposición y demuestra el enunciado.

#### Ejercicio 36

$\text{SIGUIENTES}(S) = \{\$ \}$ ;  $\text{SIGUIENTES}(A) = \{a, b, \$ \}$ ;  $\text{SIGUIENTES}(B) = \{b, \$ \}$ ;  $\text{SIGUIENTES}(C) = \{\$ \}$ ;  $\text{SIGUIENTES}(D) = \{a, b, \$ \}$ ;

$(\text{PRIMEROS}(bbD \cdot \text{SIGUIENTES}(A)) = \{b\}) \cap (\text{PRIMEROS}(a \cdot \text{SIGUIENTES}(A)) = \{a\}) = \emptyset$   
 $(\text{PRIMEROS}(a \cdot \text{SIGUIENTES}(B)) = \{a\}) \cap (\text{SIGUIENTES}(B) = \{b, \$ \}) = \emptyset$   
 $(\text{PRIMEROS}(b \cdot \text{SIGUIENTES}(C)) = \{b\}) \cap (\text{SIGUIENTES}(C) = \{\$ \}) = \emptyset$   
 $(\text{PRIMEROS}(c \cdot \text{SIGUIENTES}(D)) = \{c\}) \cap (\text{SIGUIENTES}(D) = \{a, b, \$ \}) = \emptyset$

La gramática es LL(1)

#### Ejercicio 37

$\text{SIGUIENTES}(S) = \{\$ \}$ ;  $\text{SIGUIENTES}(A) = \{e\}$ ;  $\text{SIGUIENTES}(B) = \{c, \$ \}$ ;  $\text{SIGUIENTES}(C) = \{\$ \}$ ;  $\text{SIGUIENTES}(D) = \{c, \$ \}$ ;  $\text{SIGUIENTES}(E) = \{b, c, \$ \}$ ;

$(\text{PRIMEROS}(a \cdot \text{SIGUIENTES}(A)) = \{a\}) \cap (\text{SIGUIENTES}(A) = \{e\}) = \emptyset$   
 $(\text{PRIMEROS}(c \cdot \text{SIGUIENTES}(C)) = \{c\}) \cap (\text{SIGUIENTES}(C) = \{\$ \}) = \emptyset$   
 $(\text{PRIMEROS}(B \cdot \text{SIGUIENTES}(D)) = \{c, \$ \}) \cap (\text{PRIMEROS}(bb \cdot \text{SIGUIENTES}(D)) = \{b\}) = \emptyset$   
 $(\text{PRIMEROS}(Ae \cdot \text{SIGUIENTES}(E)) = \{a, e\}) \cap (\text{SIGUIENTES}(E) = \{b, c, \$ \}) = \emptyset$

La gramática es LL(1)

**Ejercicio 38**

$\text{SIGUIENTES}(S) = \{\$ \}; \text{SIGUIENTES}(B) = \text{SIGUIENTES}(C) = \text{SIGUIENTES}(D) = \{a, c, d, \$ \};$

$(\text{PRIMEROS}(CD \cdot \text{SIGUIENTES}(B)) = \{a, c, d, \$ \}) \cap$

$(\text{PRIMEROS}(DC \cdot \text{SIGUIENTES}(B)) = \{a, c, d, \$ \}) \neq \emptyset$

$(\text{PRIMEROS}(cC \cdot \text{SIGUIENTES}(C)) = \{c\}) \cap$

$(\text{SIGUIENTES}(C) = \{a, c, d, \$ \}) \neq \emptyset$

$(\text{PRIMEROS}(Dd \cdot \text{SIGUIENTES}(D)) = \{a, c, d, \$ \}) \cap$

$(\text{PRIMEROS}(a \cdot \text{SIGUIENTES}(D) = \{a\})) \cap$

$(\text{SIGUIENTES}(D) = \{a, c, d, \$ \}) \neq \emptyset$

La gramática no es LL(1)

**Ejercicio 40a**

Supongamos que una gramática LL(1) es recursiva a izquierdas. Una gramática recursiva a izquierdas tiene reglas del tipo:

$$A \rightarrow A\alpha_1 | \dots | A\alpha_n | \beta_1 | \dots | \beta_m$$

Por ser LL(1) se debe de verificar la condición:

$$\left( \bigcap_{i=1}^n \text{primeros}(A\alpha_i \cdot \text{siguientes}(A)) \right) \cap \left( \bigcap_{j=1}^m \text{primeros}(\beta_j \cdot \text{siguientes}(A)) \right) = \emptyset$$

– Para todo  $\beta_j \neq \epsilon$  con  $j : 1 \dots m$ , se tiene:

$$\text{primeros}(\beta_j) \subseteq \text{primeros}(A) \subseteq \text{primeros}(A\alpha_i \cdot \text{siguientes}(A)) \quad \forall i : 1 \dots n.$$

$$\text{primeros}(\beta_j) \subseteq \text{primeros}(\beta_j \cdot \text{siguientes}(A))$$

y por tanto incumple la hipótesis de partida.

– Para  $\beta_k = \epsilon$ , se tiene:

$$\text{siguiente}(A) \subseteq \text{primeros}(\beta_k \cdot \text{siguientes}(A))$$

Como  $A \xRightarrow{*} \epsilon$  entonces  $\text{primeros}(\alpha_i \cdot \text{siguientes}(A)) \subseteq \text{primeros}(A\alpha_i \cdot \text{siguientes}(A))$ , y como los  $\text{primeros}(\alpha_i \cdot \text{siguientes}(A))$  son símbolos siguientes de  $A$ , se tiene que:

$$\text{siguiente}(A) \subseteq \text{primeros}(A\alpha_i \cdot \text{siguientes}(A))$$

incumpliendo también la hipótesis de partida.

**Ejercicio 40b**

Dada una gramática LL(1), supongamos que es ambigua. Si es ambigua entonces existe al menos dos secuencias de derivaciones a izquierdas posibles para una cadena dada:

$$d_1 : S \xRightarrow{*} x \quad d_2 : S \xRightarrow{*} x$$

Si se cumple esto, entonces existe una forma sentencial  $x_1 \dots x_{i-1} A \omega$  (la forma sentencial inicial  $S$ , también lo cumple) en la que a partir de ella se diferencian las dos secuencias de derivación a izquierdas  $d_1$  y  $d_2$ . Para que se cumpla esto deben existir al menos dos reglas ( $A \rightarrow \beta_1$  y  $A \rightarrow \beta_2$ ) que cumplan

$$x_i \in \text{PRIMEROS}(\beta_1 \cdot \text{SIGUIENTES}(A)) \text{ y}$$

$$x_i \in \text{PRIMEROS}(\beta_2 \cdot \text{SIGUIENTES}(A))$$

Lo cual contradice que la gramática sea LL(1)

**Ejercicio 41**  $A\alpha$

$a \in \text{PRIMEROS}(\beta \text{ SIGUIENTES}(A)).$

**Ejercicio 42**

- a)  $\text{SIGUIENTES}(A)=\{\$ \}; \text{SIGUIENTES}(B)=\{z \};$   
 $\text{SIGUIENTES}(C)=\{\$ \}; \text{SIGUIENTES}(D)=\{\$, z \};$   
 $\text{SIGUIENTES}(E)=\{x, z \};$

		w	x	y	z	\$
$(\text{PRIMEROS}(B \ C \cdot \text{SIGUIENTES}(A))=\{w, x, y, z\})$	A	r1	r1	r1	r1	
$(\text{PRIMEROS}(w \ D \cdot \text{SIGUIENTES}(B))=\{w\})$	B	r2	r3	r3	r3	
$(\text{PRIMEROS}(E \ D \cdot \text{SIGUIENTES}(B))=\{x, y, z\})$	C				r4	
$(\text{PRIMEROS}(z \ D \cdot \text{SIGUIENTES}(C))=\{z\})$	D		r5		r6	r6
$(\text{PRIMEROS}(x \ D \cdot \text{SIGUIENTES}(D))=\{x\})$	E		r8	r7	r8	
$(\text{PRIMEROS}(\text{SIGUIENTES}(D))=\{z, \$ \})$						
$(\text{PRIMEROS}(y \cdot \text{SIGUIENTES}(E))=\{y\})$						
$(\text{PRIMEROS}(\text{SIGUIENTES}(E))=\{x, z\})$						

b)

A \$	x z x \$	—
B C \$	x z z \$	1
E D C \$	x z x \$	1 3
D C \$	x z x \$	1 3 8
x D C \$	x z x \$	1 3 8 5
D C \$	z x \$	1 3 8 5
C \$	z x \$	1 3 8 5 6
z D \$	z x \$	1 3 8 5 6 4
D \$	x \$	1 3 8 5 6 4
x D \$	x \$	1 3 8 5 6 4 5
D \$	\$	1 3 8 5 6 4 5
\$	\$	1 3 8 5 6 4 5 6

**Ejercicio 43** La gramática tiene recursividad a izquierdas y problemas de factorización. Podemos proceder de dos maneras equivalentes:

- Factorizando primero y eliminando después la recursividad a izquierdas.

$$\begin{array}{lll}
 P \rightarrow P \ x & P \rightarrow P \ x & P \rightarrow y \ P^1 \ P^2 \\
 P \rightarrow y \ P \ y & P \rightarrow y \ P^1 & P^2 \rightarrow x \ P^2 \\
 P \rightarrow y \ Q & P^1 \rightarrow P \ y & P^2 \rightarrow \epsilon \\
 Q \rightarrow x & P^1 \rightarrow Q & P^1 \rightarrow P \ y \\
 Q \rightarrow z & Q \rightarrow x & P^1 \rightarrow Q \\
 & Q \rightarrow z & Q \rightarrow x \\
 & & Q \rightarrow z
 \end{array}
 \implies$$

$$\text{SIGUIENTES}(P) = \text{SIGUIENTES}(P^2) = \{y, \$ \};$$

$$\text{SIGUIENTES}(P^1) = \text{SIGUIENTES}(Q) = \{x, y, \$ \};$$

$$\text{PRIMEROS}(P \ y \cdot \text{SIGUIENTES}(P^1)) = \{y\} \cap \text{PRIMEROS}(Q \cdot \text{SIGUIENTES}(P^1)) = \{x, y\} = \emptyset$$

$$\text{PRIMEROS}(x \ P^2 \cdot \text{SIGUIENTES}(P^2)) = \{x\} \cap \text{PRIMEROS}(\text{SIGUIENTES}(P^2)) = \{y, \$ \} = \emptyset$$

$$\text{PRIMEROS}(x \cdot \text{SIGUIENTES}(Q)) = \{x\} \cap \text{PRIMEROS}(z \cdot \text{SIGUIENTES}(Q)) = \{z\} = \emptyset$$

La gramática resultante es LL(1)

- Eliminando la recursividad primero y factorizando después.

$$\begin{array}{lll}
P \rightarrow P x & P \rightarrow y P y P^a & P \rightarrow y P^b \\
P \rightarrow y P y & P \rightarrow y Q P^a & P^b \rightarrow P y P^a \\
P \rightarrow y Q & P^a \rightarrow x P^a & P^b \rightarrow Q P^a \\
Q \rightarrow x & P^a \rightarrow \epsilon & P^a \rightarrow x P^a \\
Q \rightarrow z & Q \rightarrow x & P^a \rightarrow \epsilon \\
& Q \rightarrow z & Q \rightarrow x \\
& & Q \rightarrow z
\end{array}
\Rightarrow$$

$$\text{SIGUIENTES}(P) = \text{SIGUIENTES}(P^a) = \text{SIGUIENTES}(P^b) = \{y, \$\};$$

$$\text{SIGUIENTES}(Q) = \{x, y, \$\};$$

$$\text{PRIMEROS}(P y P^a \cdot \text{SIGUIENTES}(P^b)) = \{y\} \cap \text{PRIMEROS}(Q P^a \cdot \text{SIGUIENTES}(P^b)) = \{x, z\} = \emptyset$$

$$\text{PRIMEROS}(x P^a \cdot \text{SIGUIENTES}(P^a)) = \{x\} \cap \text{PRIMEROS}(\text{SIGUIENTES}(P^a)) = \{y, \$\} = \emptyset$$

$$\text{PRIMEROS}(x \cdot \text{SIGUIENTES}(Q)) = \{x\} \cap \text{PRIMEROS}(z \cdot \text{SIGUIENTES}(Q)) = \{z\} = \emptyset$$

La gramática resultante es LL(1)

**Ejercicio 44** La gramática no es LL(1) porque es recursiva a izquierdas. Aplicando la transformación queda:

$$B \Rightarrow \{ I \} \mid i \quad I \Rightarrow B I' \quad I' \Rightarrow , B I' \mid \epsilon$$

$$\text{SIGUIENTES}(B) = \{“,”, “}”, \$\}; \text{SIGUIENTES}(I) = \text{SIGUIENTES}(I') = \{“}”\};$$

	“{”	“}”	i	“,”	\$
B	(1, “{”)		(2, i)		
I	(3, BI')		(3, BI')		
I'		(5, ε)		(4, ,BI')	

La gramática es LL(1)

S\$	{{i},{i,i}}\$	—
{I}\$	{{i},{i,i}}\$	1
BI'\$	{i},{i,i}}\$	1 3
{I}I'\$	{i},{i,i}}\$	1 3 1
BI'I'\$	i},{i,i}}\$	1 3 1 3
iI'I'\$	i},{i,i}}\$	1 3 1 3 2
}I'\$	},{i,i}}\$	1 3 1 3 2 5
,BI'\$	,{i,i}}\$	1 3 1 3 2 5 4
{I}I'\$	{i,i}}\$	1 3 1 3 2 5 4 1
BI'I'\$	i,i}}\$	1 3 1 3 2 5 4 1 3
iI'I'\$	i,i}}\$	1 3 1 3 2 5 4 1 3 2
,BI'I'\$	,i}}\$	1 3 1 3 2 5 4 1 3 2 4
iI'I'\$	i}}\$	1 3 1 3 2 5 4 1 3 2 4 2
}I'\$	}\$	1 3 1 3 2 5 4 1 3 2 4 2 5
\$	\$	1 3 1 3 2 5 4 1 3 2 4 2 5 5

**Ejercicio 46**

$$\text{PRIMEROS}(A) = \{a, b\} \cup \{c, d\} \quad \text{SIGUIENTES}(A) = \{\$\}$$

$$\text{PRIMEROS}(B) = \{a\} \cup \{\epsilon\} \quad \text{SIGUIENTES}(A) = \{b\}$$

$$\text{PRIMEROS}(C) = \{c\} \cup \{\epsilon\} \quad \text{SIGUIENTES}(A) = \{d\}$$

$$\begin{aligned}
&(\text{PRIMEROS}(\text{Bb} \cdot \text{SIGUIENTES}(\text{A})) = \{a, b\}) \cap \\
&(\text{PRIMEROS}(\text{Cd} \cdot \text{SIGUIENTES}(\text{A})) = \{c, d\}) = \emptyset \\
&(\text{PRIMEROS}(\text{aB} \cdot \text{SIGUIENTES}(\text{B})) = \{a\}) \cap \\
&(\text{SIGUIENTES}(\text{B}) = \{b\}) = \emptyset \\
&(\text{PRIMEROS}(\text{cC} \cdot \text{SIGUIENTES}(\text{C})) = \{c\}) \cap \\
&(\text{SIGUIENTES}(\text{C}) = \{d\}) = \emptyset
\end{aligned}$$

	a	b	c	d	\$
A	(1, Bb)	(1, Bb)	(2, Cd)	(2, Cd)	
B	(3, aB)	(4, ε)			
C			(5, cC)	(6, ε)	

A\$	aab\$	—
Bb\$	aab\$	1
aBb\$	aab\$	1 3
Bb\$	ab\$	1 3
aBb\$	ab\$	1 3 3
Bb\$	b\$	1 3 3
b\$	b\$	1 3 3 4
\$	\$	1 3 3 4

**Ejercicio 50** La TA LL(1) será:

	a	b	\$
S	(1, AB)	(1, AB)	(1, AB)
A	(2, aA)	(3, ε)	(3, ε)
B		(4, bB)	(5, ε)

La gramática es LL(1)

S\$	aabb\$	—
AB\$	aabb\$	1
aAB\$	aabb\$	1 2
aAB\$	abb\$	1 2 2
B\$	bb\$	1 2 2 3
bB\$	bb\$	1 2 2 3 4
bB\$	b\$	1 2 2 3 4 4
\$	\$	1 2 2 3 4 4 5

**Ejercicio 55** La gramática construida puede ser:

$$\begin{aligned}
S &\Rightarrow a S \mid A b & X &\Rightarrow c S \mid \epsilon & Z &\Rightarrow e S \\
A &\Rightarrow X Y Z \mid \epsilon & Y &\Rightarrow d S \mid \epsilon
\end{aligned}$$

**Ejercicio 59** La gramática es LL(1) ya que cumple la condición LL(1):

$$\begin{aligned}
&(\text{PRIMEROS}(\text{A x} \cdot \text{SIGUIENTES}(\text{S})) = \{x\}) \cap (\text{PRIMEROS}(\text{A} \cdot \text{SIGUIENTES}(\text{S})) = \{\$ \}) = \emptyset \\
&\text{SIGUIENTES}(\text{A}) = \{x, \$ \}
\end{aligned}$$

**Ejercicio 60**

a)



	PRIMEROS	SIGUIENTES
S	$\{u, y, z, w, x\}$	$\{\$ \}$
U	$\{u, y, z, \epsilon\}$	$\{w, x, y, z\}$
V	$\{w, x, \epsilon\}$	$\{y, z\}$
W	$\{y, z\}$	$\{v, \$ \}$

b) La TA-LL(1) (solo para los no-terminales) correspondiente es:

	x	y	z	u	v	w	\$
S	(UVW,1)	(UVW,1)	(UVW,1)	(UVW,1)		(UVW,1)	
U	( $\epsilon$ ,4)	( $\epsilon$ ,4)	( $\epsilon$ ,4)	(u,2)		( $\epsilon$ ,4)	
		(Wv,3)	(Wv,3)				
V	(xU,6)	( $\epsilon$ ,7)	( $\epsilon$ ,7)			(w,5)	
W		(y,8)	(z,9)				

c) NO, ya que la TA-LL(1) tiene entradas múltiples.

### Ejercicio 61

- a) Tiene problemas de factorización a izquierdas (S) y recursividad a izquierdas con (B).  
b) Aplicando las transformaciones de clase, queda:

$$\begin{array}{ll}
 S \rightarrow b S' & B \rightarrow b B' \\
 S' \rightarrow A b \mid B a & B' \rightarrow C B' \mid \epsilon \\
 A \rightarrow a S \mid C B & C \rightarrow c
 \end{array}$$

c) Los *primeros* y *siguientes* son,

$$\begin{array}{ll}
 \text{PRI}(b S') = \{ b \}; & \text{SIG}(S) = \{ b, \$ \}; \\
 \text{PRI}(A b) = \{ a, c \}; & \text{SIG}(S') = \{ b, \$ \}; \\
 \text{PRI}(B a) = \{ b \}; & \text{SIG}(A) = \{ b \}; \\
 \text{PRI}(a S) = \{ a \}; & \text{SIG}(B) = \{ a, b \}; \\
 \text{PRI}(C B) = \{ c \}; & \text{SIG}(B') = \{ a, b \}; \\
 \text{PRI}(b B') = \{ b \}; & \text{SIG}(C) = \{ a, b, c \}; \\
 \text{PRI}(C B') = \{ c \}; & \\
 \text{PRI}(\epsilon) = \{ \epsilon \}; & \\
 \text{PRI}(c) = \{ c \}; &
 \end{array}$$

d) La tabla de análisis LL(1), solo para los no-terminales, es:

	a	b	c	\$
S		(b S')		
S'	(A b)	(B a)	(A b)	
A	(a S)		(C B)	
B		(b B')		
B'	( $\epsilon$ )	( $\epsilon$ )	(C B')	
C			(c)	

e) Y la traza para  $b b c a$

S \$	b b c a \$	—
b S' \$	b b c a \$	1
S' \$	b c a \$	1
B a \$	b c a \$	1 3
b B' a \$	b c a \$	1 3 6
B' a \$	c a \$	1 3 6
C B' a \$	c a \$	1 3 6 7
c B' a \$	c a \$	1 3 6 7 9
B' a \$	a \$	1 3 6 7 9
a \$	a \$	1 3 6 7 9 8
\$	\$	1 3 6 7 9 8

f) Vamos a demostrarlo por reducción al absurdo: supongamos que la gramática está bien formada y existe un no-terminal,  $A \in \mathcal{N}$ , cuya fila en la Tabla de Análisis (TA) LL(1) no tiene ninguna acción derivar.

Por construcción de la TA LL(1),  $(A, x)$  producirá una acción derivar de la regla  $(A \rightarrow \alpha)$  si y solo si se cumple que  $x \in \text{PRI}(\alpha \text{ SIG}(A))$ . Por tanto, si la fila no tiene ninguna acción derivar, entonces se cumple que:

$$\text{PRI}(\alpha \text{ SIG}(A)) = \emptyset \quad \forall (A \rightarrow \alpha) \in \mathcal{P}$$

Por definición de PRI y SIG, la única posibilidad es que  $\alpha = \epsilon$  y que  $\text{SIG}(A) = \emptyset^1$ . Esto implica que A no aparece en ninguna parte derecha de la gramática, y no es axioma, lo cual nos indica que no es accesible y por tanto la gramática no estaría bien formada, invalidando la afirmación inicial.

---

<sup>1</sup>Adviértase que A no puede ser el axioma de la gramática, S, ya que  $\text{SIG}(S) = \$$

### 3.3. Análisis Sintáctico Ascendente

62. \* Dada la gramática  $\{S \rightarrow a S b \mid a b\}$ , indicad cual será el *pivote* de las siguientes formas sentenciales:  $aaabbb$  y  $aaSbb$
63. \* Para la misma gramática del ejercicio 62 describid, de una forma compacta, todos sus prefijos viables. Para el prefijo viable  $a$ , obtened todos sus ítems válidos LR(0).
64. Dado una gramática LR(0) con un ítem  $[A \rightarrow \alpha \cdot]$ , con  $\alpha \neq \epsilon$  válido para algún prefijo viable  $\gamma$ , demostrad que ningún otro ítem puede ser válido para  $\gamma$
65. Dada la siguiente gramática, calculad el conjunto de ítems LR(0) válidos para el prefijo viable  $c$ .

$$A \Rightarrow a B \mid c C \qquad B \Rightarrow b B \mid f \qquad C \Rightarrow c C \mid f$$

66. \* Dada una gramática LR(0) con un ítem válido,  $[A \rightarrow \alpha \cdot]$  con  $\alpha \neq \epsilon$ , para algún prefijo viable  $\gamma$ , demostrad que ningún otro ítem puede ser válido para  $\gamma$ .
67. \* Demostrad si la siguiente afirmación es cierta o falsa:  
Una gramática no es SLR(1) si existe un prefijo viable (para una forma sentencial a derechas) que tenga más de un ítem LR(0) válido
68. Demostrad que no pueden existir conflictos *desplazamiento/desplazamiento* en la construcción de una tabla de análisis SLR(1).
69. \* Dada una *derivación a derechas*  $S' \xRightarrow{*} \alpha A w \Rightarrow \alpha \beta w$  y siendo  $a$  el símbolo actual de la cadena de entrada. ¿Qué parte de la forma sentencial  $\alpha \beta w$  puede estar en la pila de un analizador SLR(1) y que condición se debe cumplir para que la siguiente reducción a realizar es la que aparece explícitamente en la derivación?
70. \* Justificar brevemente si las siguientes afirmaciones son ciertas o falsas:
- El autómata de la Colección Canónica de Conjuntos e Ítems LR(0) puede ser indeterminista.
  - El estado inicial del autómata de la Colección Canónica de Conjuntos e Ítems LR(0) puede aparecer en alguna otra entrada (como estado siguiente) de la tabla de análisis SLR(1).
  - La tabla de análisis (acción) SLR(1) no podrá contener ninguna columna vacía (solo acciones error).
  - La tabla de análisis (acción) SLR(1) no podrá contener ninguna fila vacía (solo acciones error).
  - Si una gramática es recursiva a izquierdas entonces no es SRL(1).

71. Dada la siguiente gramática: a) Obtened las tablas de análisis LR(0) y SLR(1);  
b) Proporcionad la traza del análisis LR(0) y SLR(0) para la cadena  $wyyz$

$$E \Rightarrow w X \mid x Y \qquad X \Rightarrow y X \mid z \qquad Y \Rightarrow y Y \mid z$$

72. Dada la gramática que se adjunta, construid la Colección Canónica de Conjuntos de ítems LR(0). ¿Es una gramática LR(0)? ¿Es SLR(1)?

$$S \Rightarrow ( A ) \qquad A \Rightarrow A , D \mid D \qquad D \Rightarrow a \mid b \mid ( A )$$

73. Razonad por qué la siguiente gramática es LL(1) pero no SLR(1):

$$S \Rightarrow A a A b \mid B b B a \quad A \Rightarrow \epsilon \quad B \Rightarrow \epsilon$$

Advertid que no basta con demostrar que es LL(1) y no es SLR(1), es necesario justificar por qué es así.

74. \* Dada la gramática que se adjunta: a) Construid la Tabla de Análisis SLR(1) a partir de la Colección Canónica de Conjuntos de ítems LR(0); b) Obtened la traza de análisis LALR(1) para la cadena  $ab$ .

$$S \Rightarrow A B \quad A \Rightarrow a A \mid \epsilon \quad B \Rightarrow b A \mid \epsilon$$

75. \* Ídem para la siguiente gramática y la cadena  $aabb$ .

$$S \Rightarrow a S A \mid \epsilon \quad A \Rightarrow B b \quad B \Rightarrow A c \mid \epsilon$$

76. \* Dada la gramática que se adjunta: a) Obtened Colección Canónica de Conjuntos de ítems LR(0); b) Construid la Tabla de Análisis LR(0) y SLR(1); c) ¿Es una gramática LR(0), y SLR(0)?;

$$S \Rightarrow A b \mid B c \quad A \Rightarrow A a \mid \epsilon \quad B \Rightarrow B a \mid \epsilon$$

77. Dada la siguiente gramática: a) Obtened la tabla de análisis SLR(1); b) Proponed una estrategia frente a los posibles conflictos, determinando la acción a seguir en cada caso, de forma que se mantenga la precedencia y la asociatividad usual entre los operadores (de mayor a menor precedencia: **no**, **y**, **o**; los operadores binarios son asociativos por la izquierda y el unario por la derecha).

$$S \Rightarrow S o S \mid S y S \mid no S \mid cierto S \mid falso \mid id$$

78. \* [J.Beaty, 1979] Demostrad que la siguiente gramática es LL(1) pero no SLR(1):

$$\begin{array}{lll} S \Rightarrow a F \mid b G & F \Rightarrow X c \mid Y d & G \Rightarrow X d \mid Y c \\ X \Rightarrow I A & Y \Rightarrow I B & I \Rightarrow \epsilon \\ A \Rightarrow \epsilon & B \Rightarrow \epsilon & \end{array}$$

79. La clase de gramáticas LL(1) y SLR(1) son incomparables:

a) [Aho et al., 1990)] Demostrad que la siguiente gramática es LL(1) pero no SLR(1):

$$S \Rightarrow A a A b \mid B b A a \quad A \Rightarrow \epsilon \quad B \Rightarrow \epsilon$$

b) Demostrad que la siguiente gramática es SLR(1) pero no LL(1):

$$S \Rightarrow S a \mid a$$

80. \* Dada la siguiente gramática:

$$\begin{array}{ll} S' \rightarrow S & \\ S \rightarrow X Z \mid X & Y \rightarrow y y Y \mid y \\ X \rightarrow x Y z & Z \rightarrow z x \end{array}$$

a) Construid la colección canónica de conjuntos de ítem LR(0).

- b) A partir de dicha colección, construid la tabla de análisis LR(0). ¿Es una gramática LR(0)? Justificad brevemente la respuesta.
- c) A partir de la misma colección canónica de conjuntos de ítem LR(0), construid la tabla de análisis SLR(1). ¿Es una gramática SLR(1)?
- d) Empleando la tabla de análisis SLR(1), proporcionad la traza de análisis SLR(1) para la cadena: **x y z z x**
- e) Dada la gramática, ¿cuál de las siguientes cadenas son prefijos viables de alguna de sus formas sentenciales a derechas? Justificad brevemente la respuesta.
- **x Y z Z**
  - **x y y y**

### 3.3.1. Ejercicios resueltos

**Ejercicio 62**       $a \ a \ \underline{a \ b} \ b \ b$        $a \ \underline{a \ S \ b} \ b$

**Ejercicio 63**       $S \mid a^+(b \mid S \ b?)$

$\{[S \rightarrow a \ . \ S \ b], [S \rightarrow a \ . \ b], [S \rightarrow \ . \ a \ S \ b], [S \rightarrow \ . \ a \ b], \}$

**Ejercicio 66**      Dado una gramática LR(0) con  $[A \rightarrow \alpha \ .]$  en un cierto estado, Si existiera otro ítem en ese estado, solo pueden pasar dos cosas:

- que sea un ítem  $[B \rightarrow \beta \ .]$ , con lo cual se producirá un conflicto reducción-reducción y no sería LR(0), o
- que sea un ítem  $[B \rightarrow \beta \ . \ \omega]$ , con lo cual se producirá un conflicto desplazamiento-reducción y no sería LR(0).

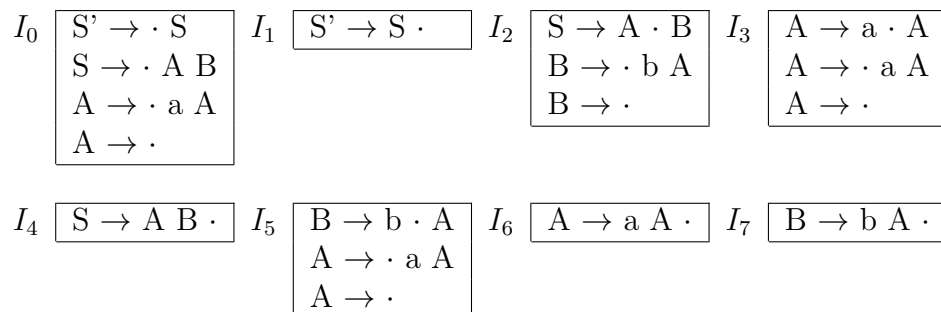
**Ejercicio 67**      Si se cumple la afirmación entonces la gramática no es LR(1) (de hecho es la definición de gramática LR(1)). Dado que las gramáticas SLR(1) son un subconjunto propio de las LR(1) entonces la gramática tampoco será SLR(1).

**Ejercicio 69**       $\alpha \ \beta$  deben estar en la pila y la condición para la reducción es que se cumpla que:  $a \in \text{siguiente}(A)$ .

**Ejercicio 70**

- a) FALSO. Por construcción la operación **sucesor** solo permite una transición por símbolo.
- b) FALSO. El estado inicial del autómata de la Colección Canónica de Conjuntos e Ítems LR(0) incorpora  $[S' \rightarrow \ . \ S]$  y dado que la gramática debe ser reducida ( $S'$  no puede aparecer en ninguna parte derecha de ninguna regla), implica que  $[S' \rightarrow \ . \ S]$  no puede ser un ítem válido para ningún otro prefijo viable.
- d) FALSO. Supondría que cuando ese símbolo terminal (válido) de la gramática aparezca como símbolo actual de la cadena de entrada, la gramática solo produce error.
- d) FALSO. De ser cierta equivaldría a tener un estado sin ningún ítem válido.
- e) FALSO. Contraejemplo, la gramática  $(A \rightarrow A \ a \mid b)$  es recursiva a izquierdas y es SLR(1).

**Ejercicio 74**



$I_0 \xrightarrow{S} I_1; I_0 \xrightarrow{A} I_2; I_0 \xrightarrow{a} I_3; I_2 \xrightarrow{B} I_4; I_2 \xrightarrow{b} I_5; I_3 \xrightarrow{A} I_6; I_3 \xrightarrow{a} I_3; I_5 \xrightarrow{A} I_7; I_5 \xrightarrow{a} I_3;$

	a	b	\$	S	A	B
0	D-3	R-3	R-3	1	2	
1			ACC			
2		D-5	R-5			4
3	D-3	R-3	R-3		6	
4			R-1			
5	D-3	R-3	R-3		7	
6		R-2	R-2			4
7			R-4			

0	ab\$	—
0a3	b\$	3
0a3A6	b\$	3 2
0A2	b\$	3 2
0A2b5	\$	3 2 3
0A2b5A7	\$	3 2 3 4
0A2B4	\$	3 2 3 4 1
0S1	\$	3 2 3 4 1

### Ejercicio 75

$I_0$	$\begin{array}{l} S' \rightarrow \cdot S \\ S \rightarrow \cdot a S A \\ S \rightarrow \cdot \end{array}$	$I_1$	$\begin{array}{l} S' \rightarrow S \cdot \end{array}$	$I_2$	$\begin{array}{l} S \rightarrow a \cdot S A \\ S \rightarrow \cdot a S A \\ S \rightarrow \cdot \end{array}$	$I_3$	$\begin{array}{l} S \rightarrow a S \cdot A \\ A \rightarrow \cdot B b \\ B \rightarrow \cdot A c \\ B \rightarrow \cdot \end{array}$
$I_4$	$\begin{array}{l} S \rightarrow a S A \cdot \\ B \rightarrow A \cdot c \end{array}$	$I_5$	$\begin{array}{l} A \rightarrow B \cdot b \end{array}$	$I_6$	$\begin{array}{l} B \rightarrow A c \cdot \end{array}$	$I_7$	$\begin{array}{l} A \rightarrow B b \cdot \end{array}$

$$I_0 \xrightarrow{S} I_1; I_0 \xrightarrow{a} I_2; I_2 \xrightarrow{S} I_3; I_2 \xrightarrow{a} I_2; I_3 \xrightarrow{B} I_5; I_3 \xrightarrow{A} I_4; I_4 \xrightarrow{c} I_6; I_5 \xrightarrow{b} I_7;$$

	a	b	c	\$	S	A	B
0	D-2	R-2		R-2	1		
1				ACC			
2	D-2	R-2		R-2	3		
3		R-5				4	5
4		R-1	D-6	R-1			
5		D-7					
6		R-4					
7		R-3	R-3	R-3			

0	aabb\$	—
0a2	abb\$	—
0a2a2	bb\$	2
0a2a2S3	bb\$	2 5
0a2a2S3B5	bb\$	2 5
0a2a2S3B5b7	b\$	2 5 3
0a2a2S3A4	b\$	2 5 3 1
0a2S3	b\$	2 5 3 1 5
0a2S3B5	b\$	2 5 3 1 5
0a2S3B5b7	\$	2 5 3 1 5 3
0a2S3A4	\$	2 5 3 1 5 3 1
0S1	\$	2 5 3 1 5 3 1

### Ejercicio 76

$I_0$	$\begin{array}{l} S' \rightarrow \cdot S \\ S \rightarrow \cdot A b \\ S \rightarrow \cdot B c \\ A \rightarrow \cdot A a \\ A \rightarrow \cdot \\ S \rightarrow \cdot B a \\ S \rightarrow \cdot \end{array}$	$I_1$	$\begin{array}{l} S' \rightarrow S \cdot \end{array}$	$I_2$	$\begin{array}{l} S \rightarrow A \cdot b \\ A \rightarrow A \cdot a \end{array}$	$I_3$	$\begin{array}{l} S \rightarrow B \cdot c \\ B \rightarrow B \cdot a \end{array}$
$I_4$	$\begin{array}{l} S \rightarrow A b \cdot \end{array}$	$I_5$	$\begin{array}{l} A \rightarrow A a \cdot \end{array}$	$I_6$	$\begin{array}{l} S \rightarrow B c \cdot \end{array}$	$I_7$	$\begin{array}{l} B \rightarrow B a \cdot \end{array}$

$$I_0 \xrightarrow{S} I_1; I_0 \xrightarrow{A} I_2; I_0 \xrightarrow{B} I_3; I_2 \xrightarrow{b} I_4; I_2 \xrightarrow{a} I_5; I_3 \xrightarrow{c} I_6; I_3 \xrightarrow{a} I_7;$$

	a	b	c	\$	S	A	B
0	$\frac{R-4}{R-6}$	R-4	R-6		1	2	3
1				ACC			
2	D-5	D-4					
3	D-7		D-6				
4				R-1			
5	R-3	R-3					
6				R-2			
7	R-5		R-5				

- No es LR(0), conflicto desplazamiento/reducción en el estado  $I_0$
- No es SLR(1), conflicto reducción/reducción en la TA SLR(1) para el par (0, a)

### Ejercicio 78

$I_0$	$S' \rightarrow \cdot S$ $S \rightarrow \cdot a F$ $S \rightarrow \cdot b G$	$I_2$	$S \rightarrow a \cdot F$ $F \rightarrow \cdot X c$ $F \rightarrow \cdot Y d$ $X \rightarrow \cdot I A$ $Y \rightarrow \cdot I B$ $I \rightarrow \cdot$	$I_7$	$X \rightarrow I \cdot A$ $Y \rightarrow I \cdot B$ $A \rightarrow \cdot$ $B \rightarrow \cdot$	...
-------	------------------------------------------------------------------------------------	-------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------	----------------------------------------------------------------------------------------------------------	-----

$$I_0 \xrightarrow{a} I_2; I_2 \xrightarrow{I} I_7; \dots$$

Dado que:

$SIGUIENTES(A) = SIGUIENTES(X) = \{c, d\}$ ; y  $SIGUIENTES(B) = SIGUIENTES(Y) = \{d, c\}$ ; en el estado  $I_7$  habrá dos conflictos reducción/reducción, por tanto NO es SLR(1).

Por otro lado, como se cumple:

$$\begin{aligned} (\text{PRIMEROS}(aF \cdot SIGUIENTES(S))=\{a\}) \cap (\text{PRIMEROS}(bG \cdot SIGUIENTES(S))=\{b\}) &= \emptyset \\ (\text{PRIMEROS}(Xc \cdot SIGUIENTES(F))=\{c\}) \cap (\text{PRIMEROS}(Yd \cdot SIGUIENTES(F))=\{d\}) &= \emptyset \\ (\text{PRIMEROS}(Xd \cdot SIGUIENTES(G))=\{d\}) \cap (\text{PRIMEROS}(Yc \cdot SIGUIENTES(G))=\{c\}) &= \emptyset \end{aligned}$$

entonces SI que es LL(1).

### Ejercicio 80

a)	$I_0$	$S' \rightarrow \cdot S$ $S \rightarrow \cdot X Z$ $S \rightarrow \cdot X$ $X \rightarrow \cdot x Y z$	$I_1$	$S' \rightarrow S \cdot$	$I_2$	$S \rightarrow X \cdot Z$ $S \rightarrow X \cdot$ $Z \rightarrow \cdot z x$	$I_3$	$X \rightarrow x \cdot Y z$ $Y \rightarrow \cdot y y Y$ $Y \rightarrow \cdot y$
	$I_4$	$S \rightarrow X Z \cdot$	$I_5$	$Z \rightarrow z \cdot x$	$I_6$	$X \rightarrow x Y \cdot z$	$I_7$	$Y \rightarrow y \cdot y Y$ $Y \rightarrow y \cdot$
	$I_8$	$Z \rightarrow z x \cdot$	$I_9$	$X \rightarrow x Y z \cdot$	$I_{10}$	$Y \rightarrow y y \cdot Y$ $Y \rightarrow \cdot y y Y$ $Y \rightarrow \cdot y$	$I_{11}$	$Y \rightarrow y y Y \cdot$

$$\begin{aligned} I_0 \xrightarrow{S} I_1 \quad I_0 \xrightarrow{X} I_2 \quad I_0 \xrightarrow{x} I_3 \quad I_2 \xrightarrow{Z} I_4 \quad I_2 \xrightarrow{z} I_5 \quad I_3 \xrightarrow{Y} I_6 \\ I_3 \xrightarrow{y} I_7 \quad I_5 \xrightarrow{x} I_8 \quad I_6 \xrightarrow{z} I_9 \quad I_7 \xrightarrow{y} I_{10} \quad I_{10} \xrightarrow{Y} I_{11} \quad I_{10} \xrightarrow{y} I_7 \end{aligned}$$

b) Tabla de Análisis LR(0)



	ACC	SUCC						
		x	y	z	S	X	Y	Z
0	D	3			1	2		
1	A							
2	D/R-2			5				4
3	D		7				6	
4	R-1							
5	D	8						
6	D			9				
7	D/R-5		10					
8	R-6							
9	R-3							
10	D		7				11	
11	R-4							

La gramática no es LR(0) ya que tiene entradas múltiples en los estados 2 y 7.

c) Tabla de Análisis SLR(1)

	x	y	z	\$	S	X	Y	Z
0	D-3				1	2		
1				Acc				
2			D-5	R-2				4
3		D-7					6	
4				R-1				
5	D-8							
6			D-9					
7		D-10	R-5					
8				R-6				
9			R-3	R-3				
10		D-7					11	
11			R-4					

La gramática es SLR(1) ya que no tiene entradas múltiples en su tabla de análisis.

d) La traza para la cadena  $x y z z x$

0	x	y	z	z	x	\$	—
0 x 3			y	z	z	x \$	—
0 x 3 y 7				z	z	x \$	—
0 x 3 Y 6				z	z	x \$	5
0 x 3 Y 6 z 9					z	x \$	5
0 X 2					z	x \$	5 3
0 X 2 z 5						x \$	5 3
0 X 2 z 5 x 8							\$ 5 3
0 X 2 Z 4							\$ 5 3 6
0 S 1							\$ 5 3 6 1

e)  $[x y y y]$  si que es un prefijo viable ya que se corresponde al camino  $I_0 - I_3 - I_7 - I_{10} - I_7$  en la colección canónica de conjuntos de ítem LR(0), siendo el  $I_7$  el conjunto de ítem válidos para este prefijo viable.

Por la misma razón,  $[x Y z Z]$  no es un prefijo viable ya que no existe un camino desde  $I_0$  que lo recorra.

## 4. Análisis Semántico

### 4.1. Gramáticas de Atributos

81. Dada la siguiente gramática: a) Construid una gramática S-atribuida que calcule el número de  $a$  y de  $b$  que tiene una frase cualquiera del lenguaje generado por dicha gramática; y b) obtened obtener la traza de análisis ascendente y evaluación de los atributos para la frase:  $(a, (b))$

$$S \Rightarrow (A) \quad A \Rightarrow A, D \mid D \quad D \Rightarrow (A) \mid a \mid b$$

82. \* Explicad las restricciones que ha de cumplir una gramática de atributos para que sea L-Atribuida, y la relación de éstas con las gramáticas S-Atribuidas.
83. \* Construid un Esquema de Traducción Dirigida por la Sintaxis que, para la siguiente gramática, devuelva la cantidad de  $a$ ,  $b$  y  $c$  que se produce en el análisis de una cadena.

$$S \Rightarrow a S A \mid \epsilon \quad A \Rightarrow B b \quad B \Rightarrow A c \mid \epsilon$$

84. \* La siguiente gramática corresponde a la especificación sintáctica de listas válidas de PROLOG:

$$\text{Lista} \Rightarrow [] \mid [T] \quad T \Rightarrow T, T \mid \text{Lista} \mid id$$

Diseñad una gramática atribuida capaz de procesar una lista y averiguar su longitud y extensión. Se define longitud de una lista o término, como el número de elementos que la componen. Análogamente, se define extensión de una lista o término, como la longitud máxima de cualquiera de sus elementos. Por ejemplo, la lista

$[ [a1, a2], a3, [a4, [a5, a6], [a7], a8] ]$  tiene longitud 3 y extensión 4.

85. Construid una gramática S-atribuida que obtenga el número binario correspondiente al complemento a dos de otro número, basándose en la siguiente gramática. Las reglas semánticas deben definirse únicamente en función de los operadores aritméticos decimales básicos y de la operación de concatenación “.”

$$S \Rightarrow C \quad C \Rightarrow 0 C \mid 1 C \mid 0 \mid 1$$

86. Dada la siguiente gramática que genera árboles binarios de números enteros (en forma linealizada):

$$S \rightarrow A \\ A \rightarrow (num A A) \mid (num)$$

Definamos el nivel de profundidad de un nodo en un árbol binario como el número mínimo de ramas que hay que recorrer para ir del nodo raíz hasta este nodo. Diseñad un ETDS para esta gramática que obtenga el nivel de profundidad que ocupa el número entero de mayor valor; en caso de que el valor máximo se repita, se debe indicar la profundidad mínima. Ejemplo, para la cadena  $(2(1(0)(1))(3(2)(3)))$  la contestación sería: valor máximo 3 y profundidad 1.

87. \* Diseñad un ETDS para traducir declaraciones de objetos de un lenguaje (parecido al) C a un lenguaje (parecido al) PASCAL, mediante la siguiente gramática:

$$S \rightarrow T V \quad V \rightarrow id D \quad D \rightarrow [cte] D \mid \epsilon \quad T \rightarrow int \mid float \mid char$$

Ejemplos:

<code>int A [7];</code>	$\Rightarrow$	<code>var A: array[0..6] of integer;</code>
<code>float B [6] [5];</code>	$\Rightarrow$	<code>var B: array[0..5,0..4] of real;</code>
<code>char C;</code>	$\Rightarrow$	<code>var A: char;</code>

88. \* Dada la siguiente gramática, diseñad un ETDS que permita calcular el número de pares de paréntesis apertura-cierre y el número de *id* de cualquier cadena de entrada.

$$L \rightarrow ( L ) \mid L B \mid \epsilon \quad B \rightarrow id$$

89. \* Diseñad un ETDS para traducir declaraciones de objetos de un lenguaje (parecido al) C a un lenguaje (parecido al) PASCAL, mediante la siguiente gramática:

$$D \rightarrow T id ; \quad T \rightarrow int \mid float \mid char \mid struct \{ C \} \quad C \rightarrow C D \mid D$$

El resultado de la traducción puede acumularse en un atributo `trad` de tipo cadena. Por ejemplo:

<code>int a;</code>	$\Rightarrow$	<code>a : integer;</code>
<code>struct { float b; char c; } r;</code>	$\Rightarrow$	<code>r: record b: real; c: char; end;</code>

90. \* Para la siguiente gramática, construid un ETDS que calcule el valor numérico (y lo asocie a un atributo del no-terminal E) de una expresión matemática definida por la gramática.

$$E \rightarrow T R \quad T \rightarrow cte \mid ( E ) \quad R \rightarrow + T R \mid - T R \mid \epsilon$$

91. \* Para la misma gramática del ejercicio 92, construid un ETDS que transforme una expresión de notación infija a postfija. Por ejemplo:  $2 + 3 - 4 \Rightarrow 2 3 + 4 -$  o este otro  $2 + ( 3 - 4 ) \Rightarrow 2 3 4 - +$

92. \* Para la siguiente gramática, construid un ETDS que calcule el valor numérico decimal de un número binario. Por ejemplo, el valor decimal de 110 es 6.

$$S \rightarrow 0 S \mid 1 S \mid 0 \mid 1$$

93. \* Dada la siguiente gramática, que genera cadenas que representan expresiones matemáticas, construid un ETDS que permita obtener como resultado una cadena que sea la derivada de la cadena de entrada. Por ejemplo:  $x*x+\sin(x) \Rightarrow 2*x*d_x+\cos(x)*d_x$

$$E \rightarrow E + T \mid T \quad T \rightarrow T * F \mid F \quad F \rightarrow \sin ( E ) \mid \cos ( E ) \mid ( E ) \mid x \mid cte$$

94. \* Dada la siguiente gramática, que permite representar árboles binarios, se pide: a) construir un ETDS que obtenga el máximo y el mínimo valor numérico que aparezcan en el árbol; y b) sobre el ETDS del apartado anterior, añadid nuevas acciones semánticas para comprobar si el árbol está ordenado. Un árbol está ordenado si el valor numérico de cada nodo es mayor o igual que todos los valores numéricos del subárbol izquierdo y menor o igual que todos los del subárbol derecho. Por ejemplo, este el árbol está ordenado (5 (2 nil nil) (7 nil nil)); sin embargo, este otro no lo está (4 (2 (1 nil nil) (7 nil nil)) (10 nil nil)).

$$A \rightarrow ( \text{cte } A \ A ) \mid \text{nil}$$

#### 4.1.1. Ejercicios resueltos

##### Ejercicio 82

Dado una gramática de atributos  $GA = (G, A, R)$ , donde  $G$  es una gramática incontextual,  $A = \cup_{x \in (N \cup T)} A(x)$  el conjunto de atributos y  $R = \cup_{k \in P} R(k)$  el conjunto de acciones semánticas,  $GA$  será **L-atribuida**, si  $\forall (k : X_0 \rightarrow X_1 X_2 \dots X_n) \in P$ , los atributos heredados  $X_i.a$  ( $1 \leq i \leq n \wedge a \in A(X_i)$ ) con  $(X_i.a := f(X_r.b, \dots, X_s.c)) \in R(k)$  son evaluados en términos de:

- atributos de  $X_1, X_2, \dots, X_{i-1}$  ó
- atributos heredados de  $X_0$ .

Mientras que,  $GA$  será **S-atribuida**, si todos los atributos empleados en todas las acciones semánticas son sintetizados. Atendiendo a la definición anterior, toda gramática S-atribuida es L-atribuida.

##### Ejercicio 83

S	$\Rightarrow a \ S \ A$	S.a = S'.a + 1; S.b = S'.b + A.b; S.c = S'.c + A.c;
	$\Rightarrow \epsilon$	S.a = S.b = S.c = 0;
A	$\Rightarrow B \ b$	A.b = B.b + 1; A.c = B.c
B	$\Rightarrow A \ c$	B.b = A.b; B.c = A.c + 1
	$\Rightarrow \epsilon$	B.b = B.c = 0;

##### Ejercicio 84

L	$\Rightarrow [ ]$	L.l = L.e = 0;
	$\Rightarrow [ T ]$	L.l = T.l; L.e = T.e;
T	$\Rightarrow T^1, T^2$	T.l = T <sup>1</sup> .l + T <sup>2</sup> .l; T.e = max{T <sup>1</sup> .e, T <sup>2</sup> .e};
	$\Rightarrow L$	T.l = 1; T.e = L.l
	$\Rightarrow \text{id}$	T.l = T.e = 1;

### Ejercicio 87

$S \Rightarrow T V$	$S.s := V.s \odot T.s$
$V \Rightarrow id D$	$\underline{\text{si}} D.s = nil \text{ entonces } V.s := \text{"var"} \odot id.nom \odot \text{"."}$ $\underline{\text{sino}} V.s := \text{"var"} \odot id.nom \odot \text{" array[ " } \odot D.s$
$D \Rightarrow [ cte ] D$	$\underline{\text{si}} D_1.s = nil \text{ entonces } D.s := \text{"0.."} \odot cte.num - 1 \odot \text{" ] of"}$ $\underline{\text{sino}} D.s := \text{"0.."} \odot cte.num - 1 \odot \text{" , " } \odot D_1.s$
$\Rightarrow \epsilon$	$D.s := nil;$
$T \Rightarrow int$	$T.s := \text{"integer"};$
$\Rightarrow float$	$T.s := \text{"real"};$
$\Rightarrow char$	$T.s := \text{"char"};$

### Ejercicio 88

$L \rightarrow (L)$	$L.n \leftarrow L_1.n; L.p \leftarrow L_1.p + 1;$
$L \rightarrow LB$	$L.n \leftarrow L_1.n + B.n; L.p \leftarrow L_1.p;$
$L \rightarrow \epsilon$	$L.n \leftarrow 0; L.p \leftarrow 0;$
$B \rightarrow id$	$B.n \leftarrow 1;$

### Ejercicio 89

$D \rightarrow T id ;$	$D.trad = id.nom \oplus \text{" : " } \oplus T.trad \oplus \text{" , "}$
$T \rightarrow int$	$T.trad = \text{"integer"}$
$\rightarrow float$	$T.trad = \text{"real"}$
$\rightarrow char$	$T.trad = \text{"char"}$
$\rightarrow struct \{ C \}$	$T.trad = \text{"record"} \oplus C.trad \oplus \text{"end"}$
$C \rightarrow C D$	$C.trad = C'.trad \oplus D.trad$
$\rightarrow D$	$C.trad = D.trad$

### Ejercicio 90

$E \rightarrow T$	$R.h = T.s$
$R$	$E.s = R.s$
$R \rightarrow + T$	$R^1.h = R.h + T.s$
$R$	$R.s = R^1.s$
$R \rightarrow - T$	$R^1.h = R.h - T.s$
$R$	$R.s = R^1.s$
$R \rightarrow \epsilon$	$R.s = R.h$
$T \rightarrow ( E )$	$T.s = E.s$
$T \rightarrow cte$	$T.s = LEXVAL(cte)$

### Ejercicio 91

$E \rightarrow T$	$R.h = T.s$
$R$	$E.s = R.s$
$R \rightarrow + T$	$R^1.h = R.h \odot T.s \odot +$
$R$	$R.s = R^1.s$
$R \rightarrow - T$	$R^1.h = R.h \odot T.s \odot -$
$R$	$R.s = R^1.s$
$R \rightarrow \epsilon$	$R.s = R.h$
$T \rightarrow ( E )$	$T.s = E.s$
$T \rightarrow cte$	$T.s = cte$

### Ejercicio 92

$S \rightarrow 0 S$	$S.n = S^1.n + 1; \quad S.r = 0 + S^1.r;$
$S \rightarrow 0 S$	$S.n = S^1.n + 1; \quad S.r = 2^{S.n} + S^1.r;$
$S \rightarrow 0$	$S.n = 0; \quad S.r = 0;$
$S \rightarrow 1$	$S.n = 0; \quad S.r = 1;$

### Ejercicio 93

$E \rightarrow E + T$	$E.f = E^1.f \oplus T.f; \quad E.d = E^1.d \oplus T.d;$
$E \rightarrow T$	$E.f = T.f; \quad E.d = T.d;$
$T \rightarrow T * F$	$T.f = T^1.f \otimes F.f; \quad T.d = T^1.d \otimes F.f \oplus T^1.f \otimes F.d;$
$T \rightarrow F$	$T.f = F.f; \quad T.d = F.d;$
$F \rightarrow \text{sen}( E )$	$F.f = \text{sen}( E.f ); \quad F.d = \cos( E.f ) \otimes E.d;$
$F \rightarrow \cos( E )$	$F.f = \cos( E.f ); \quad F.d = -\text{sen}( E.f ) \otimes E.d;$
$F \rightarrow ( E )$	$F.f = ( E.f ); \quad F.d = ( E.d );$
$F \rightarrow x$	$F.f = x; \quad F.d = d_x;$
$F \rightarrow cte$	$F.f = cte; \quad F.d = 0;$

### Ejercicio 94

$A \rightarrow ( cte \ A \ A )$	$A.max = \max \{ cte.val, A^1.max, A^2.max \}$
	$A.min = \min \{ cte.val, A^1.min, A^2.min \}$
	$A.ord = (A^1.ord \ \&\& \ A^2.ord \ \&\& \ (A^1.max \leq cte.val \leq A^1.min))$
$A \rightarrow nil$	$A.max = -\infty; \ A.min = \infty; \ A.ord = true;$

## 4.2. Comprobación de Tipos

Colección de ejercicios que complementan el *caso de estudio* estudiado en las clases de teoría.

95. Explicad que se entiende por conversión de tipos explícita e implícita, quien la realiza y poned un ejemplo de cada una de ellas.
96. Explicad la diferencia entre la equivalencia de tipos por nombre y estructural.
97. Explicad las diferencias entre el análisis de tipos estático y el dinámico. Poned un ejemplo de error detectado por cada uno de ellos.
98. \* Considerando que se ha completado la fase de declaración de los objetos, diseñad un ETDS para la comprobación semántica de tipos en el par de reglas:

$$E \rightarrow * \text{ id} \quad \text{y} \quad E \rightarrow \text{id}$$

99. \* Construid un ETDS para la siguiente gramática: **a)** Desarrollad la declaración de tipos, introduciendo en la tabla de símbolos cada identificador que aparezca en  $L_{nombres}$  tantas veces como tipos distintos tenga asociados en  $L_{tipos}$ . El significado de la regla “ $D \rightarrow L_{tipos} ( D )$ ” es que a todos los nombres mencionados en la declaración  $D$ , dentro del paréntesis, se les dan los tipos que aparecen en  $L_{tipos}$ , independientemente de cuantos niveles de anidamiento existan. **b)** Realizad la comprobación de tipos para las instrucciones y expresiones.

$$\begin{array}{ll} S & \rightarrow D ; I \\ D & \rightarrow L_{tipos} L_{nombres} \mid L_{tipos} ( D ) \mid D D \\ L_{tipos} & \rightarrow T L_{tipos} \mid T \\ T & \rightarrow entero \mid real \mid complejo \\ L_{nombres} & \rightarrow id \mid id L_{nombres} \\ I & \rightarrow id := E \\ E & \rightarrow id \mid E + E \mid ( E , E ) \mid cte \mid ctr \end{array}$$

100. \* Dada la siguiente gramática que genera declaraciones para un único identificador, obtened un ETDS que asigne a cada identificador su lista de opciones y que no permita opciones repetidas.

$$\begin{array}{ll} S & \rightarrow id \text{ Lopciones} \\ \text{Lopciones} & \rightarrow \text{Lopciones Opciones} \mid \epsilon \\ \text{Opciones} & \rightarrow \text{Modo} \mid \text{Escala} \mid \text{Precision} \mid \text{Base} \\ \text{Modo} & \rightarrow real \mid complejo \\ \text{Escala} & \rightarrow fija \mid flotante \\ \text{Precision} & \rightarrow simple \mid doble \\ \text{Base} & \rightarrow binaria \mid decimal \end{array}$$

101. Dada la gramática, diseñad un ETDS que permita realizar la declaración de los objetos y las comprobaciones semánticas correspondientes, teniendo en cuenta que: **a)** el operador  $+$  está definido para *cadena*s (concatenación) y para *enteros* (suma); **b)**  $a(p, f)$  es una subcadena de la cadena  $a$  determinada por los índices  $p$  y  $f$ ; y **c)**  $|a|$  es la longitud (entera) actual de la cadena  $a$ .

$$\begin{aligned}
P &\rightarrow D ; S \\
D &\rightarrow D ; D \mid id : T \\
T &\rightarrow entero \mid cadena ( num ) \\
S &\rightarrow id := E \\
E &\rightarrow E + E \mid id ( E , E ) \mid id
\end{aligned}$$

102. \* La siguiente gramática permite declarar identificadores de tipo *lista* y obtener expresiones de pertenencia de un elemento a una lista. Desarrollad un ETDS que permita hacer una comprobación de tipos de forma que: **a)** los elementos de una lista siempre sean del mismo tipo; y **b)** la relación de pertenencia sea compatible con la condición (a)

$$\begin{aligned}
P &\rightarrow D ; S \\
P &\rightarrow D ; D \mid id : T \\
T &\rightarrow char \mid list\ of\ T \\
S &\rightarrow E\ en\ E \\
E &\rightarrow literal \mid id \mid ( L ) \\
L &\rightarrow E , L \mid E
\end{aligned}$$

#### 4.2.1. Ejercicios resueltos

##### Ejercicio 98

$E \Rightarrow id$	$\underline{Si} \neg [ObtenerTDS(id.nom, E.t)] \{ E.t=terror; MenError(.); \}$
$E \Rightarrow * id$	$\underline{Si} \neg [ObtenerTDS(id.nom, id.t) \wedge id.t=tpuntero(E.t)]$ $\{ E.t=terror; MenError(.); \}$

##### Ejercicio 99

$S \rightarrow D ; I$	
$D \rightarrow L\ N$	$D.n = N.n; \forall x \in N.n: insertarConjTipos(x, L.t);$
$\rightarrow \overline{L} ( D )$	$D.n = D'.n; \forall x \in D'.n: insertarConjTipos(x, L.t);$
$\rightarrow \overline{D\ D}$	$D.n = D^1 \cup D^2;$
$L \rightarrow T\ L$	$L.t = L'.t \cup T.t;$
$\rightarrow \overline{T}$	$L.t = T.t;$
$T \rightarrow entero$	$T.t = \{tentero\};$
$\rightarrow real$	$T.t = \{treal\};$
$\rightarrow complejo$	$T.t = \{tcomplejo\};$
$N \rightarrow id\ N$	$N.n = N'.n \cup \{id.nom\};$
$\rightarrow \overline{id}$	$N.n = \{id.nom\};$
$I \rightarrow id = E$	$id.lt = buscaTipos(id.nom); \underline{Si} (id.lt = \emptyset) MenError(...);$
$\rightarrow$	$\underline{Si} (id.lt \cap E.lt = \emptyset) MenError(...);$
$E \rightarrow E + E$	$E.lt = E^1.lt \cap E^2.lt; \underline{Si} (E.lt = \emptyset) MenError(...);$
$\rightarrow id$	$E.lt = buscaTipos(id.nom); \underline{Si} (E.lt = \emptyset) MenError(...);$
$\rightarrow ( E , E )$	$E.lt = \{tcomplejo\};$
$\rightarrow cte$	$E.lt = \{tentero\};$
$\rightarrow ctr$	$E.lt = \{treal\};$



## Ejercicio 100

$S \rightarrow id\ L$	<code>insertarListaOpciones(id.nom, L.m, L.e, L.p, L.b);</code>
$L \rightarrow L\ O$	<code>L.m = L'.m <math>\cup</math> O.m; L.e = L'.e <math>\cup</math> O.e; L.p = L'.p <math>\cup</math> O.p; L.b = L'.b <math>\cup</math> O.b;</code>
$\rightarrow \epsilon$	<code>L.m = L.e = L.p = L.b = 0;</code>
$O \rightarrow M$	<code>O.m = M.t; O.e = O.p = O.b = 0;</code>
$\rightarrow \overline{E}$	<code>O.e = E.t; O.m = O.p = O.b = 0;</code>
$\rightarrow \overline{P}$	<code>O.p = P.t; O.m = O.e = O.b = 0;</code>
$\rightarrow \overline{B}$	<code>O.b = B.t; O.m = O.e = O.p = 0;</code>
$M \rightarrow real$	<code>M.t = treal;</code>
$\rightarrow complejo$	<code>M.t = tcomplejo;</code>
$E \rightarrow fija$	<code>E.t = tfija;</code>
$\rightarrow flotante$	<code>E.t = tflotante;</code>
$P \rightarrow simple$	<code>P.t = tsimple;</code>
$\rightarrow doble$	<code>P.t = tdoble;</code>
$B \rightarrow decimal$	<code>B.t = tdecimal;</code>
$\rightarrow binaria$	<code>B.t = tbinaria;</code>

## Ejercicio 102

$P \rightarrow D ; S$	
$\rightarrow \overline{D} ; D$	
$\rightarrow id : T$	<code>insertarTDS(id.nom, T.t);</code>
$T \rightarrow list\_of\ T$	<code>T.t = tlista(T'.t);</code>
$\rightarrow char$	<code>T.t = tcarácter;</code>
$S \rightarrow E\ en\ E$	<code><math>\underline{Si} \neg [ (E^2.t = tlista(E^2.tel)) \wedge (E^1.t = E^2.tel) ]\ MenError(...);</math></code>
$E \rightarrow ( L )$	<code>E.t = tlista(L.t);</code>
$\rightarrow literal$	<code>E.t = tcarácter;</code>
$\rightarrow id$	<code><math>\underline{Si} \neg [ obtenerTDS(id.nom, E.t) ]\ MenError(...);</math></code>
$L \rightarrow E , L$	<code>L.t = L'.t; <math>\underline{Si} (L'.t \neq E.t)\ MenError(...);</math></code>
$\rightarrow \overline{E}$	<code>L.t = E.t;</code>

## 5. Gestión de Memoria

103. Dado el siguiente fragmento de un programa en C:

```
int a[5] ;
real resultado ;
int imprime (int valor) {
    printf("El resultado es %d\n", valor);
}
int sum (int m) {
    int temporal // MOSTRAR PILA
    if (m < 1) return 0;
    else {
        temporal = m+sum(m-1);
        return temporal ;
    }
}
int main() { // MOSTRAR TDS
    a[1] = 2 ;
    resultado = sum (a[1]);
    imprime (resultado);
}
```

Se pide:

- Suponiendo que la TDS está organizada en forma de pila, determinad su estado al llegar al punto marcado como MOSTRAR TDS.
- Indicad el estado de la pila de ejecución, calculando el valor de los parámetros y de las variables locales, cada vez que la ejecución del programa pase por el punto marcado como MOSTRAR PILA.

104. Dado el siguiente fragmento de un programa en C, y suponiendo que la talla de los enteros es 1 y de los reales 2 :

```
|-----|
| int A (float p, int q, int r) |
| { float x, y;                // <== TDS |
|   return p+q+r; }            // <== Pila |
|                               |
| int B (int p, float q)       |
| { float x, y;                // <== TDS |
|   if (p == q) return A(q, p, 2); |
|   else B(p-1, q); }         |
|                               |
| int main()                   |
| { float x = 0.0, int y = 1;   // <== TDS |
|   printf("%d", B(y, x)); }    |
|-----|
```

- Mostrad el estado de la TDS, con todas sus tablas auxiliares, en el punto de control **TDS** durante el proceso de compilación.
- Mostrad el estado de la pila de Registros de Activación (*frames*) cada vez que la ejecución del programa pase por el punto de control **Pila**.

105. \* Dado el siguiente fragmento de un programa en C:

```
int fib[10] ;
int i= 0 ;
void printint (int x) {
    printf("%d\n", x);
}
int cal_fibo(int x) {
    if(x<2) return x;
    else                                     // MOSTRAR PILA
        return cal_fibo(x-1) + cal_fibo(x-2);
}
int main() {
    int num;

                                                // MOSTRAR TDS

    num = 3 ;
    fib[num] = cal_fibo(num);
    for(i=0; i<10 ; i++) printint(fib[i]);
}
```

Se pide:

- Suponiendo que la Tabla de Símbolos está gestionada como una pila, determinad su contenido, indicando el valor de todos sus campos, al llegar al punto marcado como MOSTRAR TDS.
- Indicad el contenido de la pila de ejecución, calculando el valor de los parámetros y de las variables locales, cada vez que la ejecución del programa pase por el punto marcado como MOSTRAR PILA. No olvidad indicar el valor del **frame pointer**.

106. \* Dado el siguiente fragmento de un programa en C, y suponiendo que la talla de los enteros es 1, la de los reales 2 y la del segmento de gestión (enlaces de control y dirección de retorno) 2,

```
|-----|
| float max[150];                                |
| |                                                |
| int f1 (int p1, int p2)                        |
| { float v1, v2;                                |
|   if (p1 <= p2) v1 = 1;                        // <== Foto: TDS |
|   else {                                        |
|       v2 = p1 -1;                             // <== Foto: Pila |
|       v1= f1(v2,p2) * p1;                      |
|   }                                            |
|   return v1 ; }                               |
| |                                                |
| int main()                                    |
| { float n;                                    |
|   max[1] = n = 4;                             // <== Foto: TDS |
|   printf("%d", f1(max[1], 2));                 |
| |                                                // <== Foto: Pila |
| }                                              |
|-----|
```

- a) Mostrad el estado de la TDS, con todas sus tablas auxiliares, en el punto de control **Foto: TDS** durante el proceso de compilación.
- b) Mostrad el estado de la pila de Registros de Activación (*frames*) cada vez que la ejecución del programa pase por el punto de control **Foto: Pila**.

107. Dado el siguiente fragmento de un programa en PASCAL:

```

program prueba;
var a: record a1: real; a2: real end;
    b: array [0..9, 0..99] of integer;
    c: integer;
function P (var x,y: real): integer;
var c: integer;
    fuction Q: integer;
begin
    ....
end;
begin
    ...
    if x >= y then c := P(x,y+1)
    else c := Q;
    ...
end;
procedure R (var x,y: real);
var c: integer;
begin
    ...
    c := P(x,y);
    ...
end;
begin
    ...
    R(4,3);
    ...
end.

```

Mostrad el estado de la TDS en el momento de la compilación del punto de control **A**. Igualmente representad el estado de la pila de Registros de Activación en el proceso de ejecución del punto de control **B**. Considerad que la talla de los enteros es 2 y la de los reales es 4.

108. Contestad brevemente a las siguientes cuestiones:

- a) ¿Qué acciones se deben realizar para comprobar que el tipo de los parámetros actuales de un llamada a una función coincide con el de los parámetros formales?
- b) ¿Cómo se reserva espacio para el valor de retorno de una función?
- c) Dónde se almacena la información relativa a la primera instrucción del segmento de código de una función?
- d) ¿Cómo se reserva espacio en un Registro de Activación para las variables temporales?

109. \* Dado el siguiente fragmento de un programa en C. Suponiendo que la talla de enteros es 2 y la de los reales, enlaces de control y direccion de retorno es 4,

```

-----
#include<stdio.h>
int a ;

int f1(int p1, float p2, int p3) {
    float b ;

    b = p2 * p1 +p3 ;      // <--- PUNTO DE CONTROL

    return b ; }

int main () {
    int c = 2 ;

    printf( "%d\n", f1(c, c*2, c*3)); }
-----

```

- Mostrad el contenido completo de la TDS en el PUNTO DE CONTROL.
- Indicad el desplazamiento relativo de los parámetros `p1` y `p2` que aparecen en la instrucción del PUNTO DE CONTROL.
- Mostrad el contenido de los frames (RA) que se encuentran en la pila de ejecución cada vez que se pasa por el PUNTO DE CONTROL.

110. \* Dado el siguiente programa C:

```

-----
float f1(int x, int y)
{
    if (x>y) return f1(x-1,y)
    else return x+y;          <=== A
}
void main ()
{ int a = 3; float c; int b = 2;
  { int a = 6;
    c = a / 2 * b;           <=== B
  }
  { int a = 2; float b = 1.0
    c = a / 2 * b;          <=== C
  }
  c = f1(a, b);
}
-----

```

- Suponiendo que la talla de enteros es 2 y la de los reales es 4, mostrad el contenido completo de la TDS en el punto de control **C**.
- Indicad el desplazamiento relativo de las 4 variables de la instrucción `c = a/2 * b` en los dos puntos de control: **B**, **C**.
- Mostrad el contenido (en términos de Registros de Activación) del estado de la pila de ejecución en el punto de control **A** (antes del `return`).

111. \* Dado el siguiente programa C, y suponiendo que la talla de enteros es 2, la de los reales es 4 y la talla del segmento de enlaces de control es 8, mostrad el contenido completo de la TDS en los puntos de control (1) y (2).

```

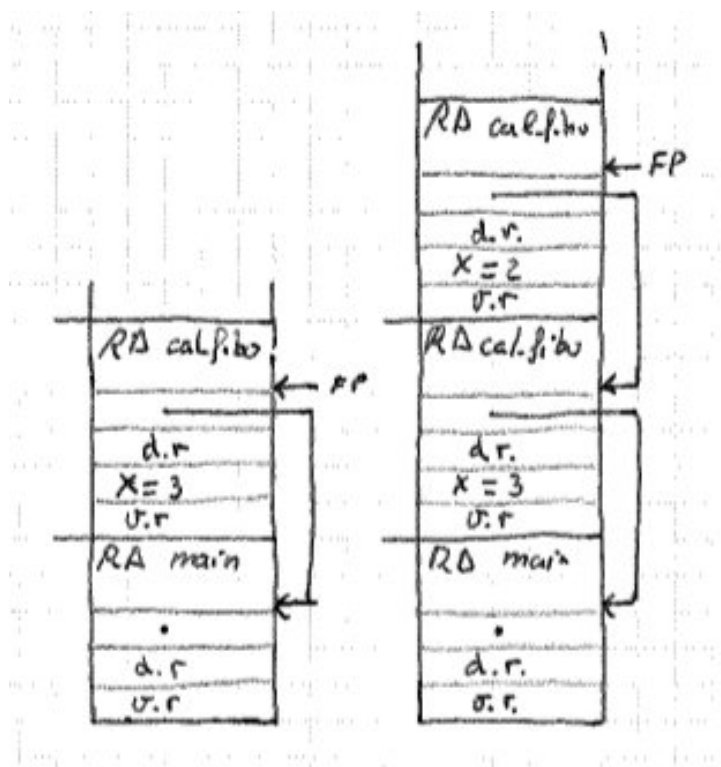
-----
float G (int a, float b)
{ float i=3.0;
  ...
  return b * i;          // <---- (1)
}
struct{int c1; float c2; int c3} x;
int mem;
}
void main ()
{ int i=2, A[10]; float k=5.0;
  ...
  printf("%f\n", G(i, k));    // <---- (2)
}
-----

```

## 5.1. Ejercicios resueltos

### Ejercicio 105

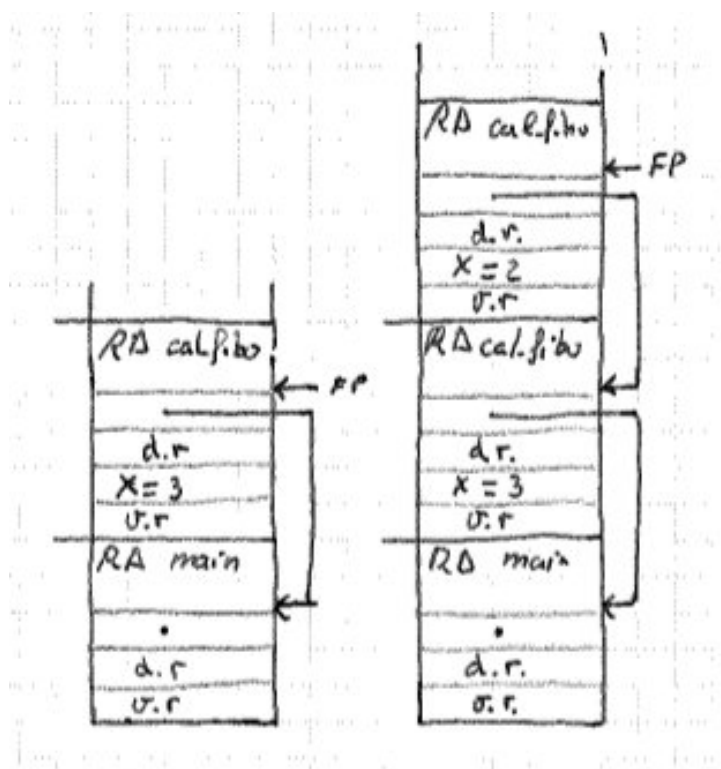
	nom		n	$\delta$	tipo	Subtablas
1	fib	var_global	0	0	tvector	$\Rightarrow$ (10, tentero)
2	i	var_global	0	10	tentero	
3	printint	funcion	0	—	tfuncion	$\Rightarrow$ (tentero) $\rightarrow$ tvacio
4	cal_fibo	funcion	0	—	tfuncion	$\Rightarrow$ (tentero) $\rightarrow$ tentero
5	main	funcion	0	—	tfuncion	$\Rightarrow$ (tvacio) $\rightarrow$ tentero
6	num	var_local	1	0	tentero	



### Ejercicio 106

	nom		n	$\delta$	tipo	Subtablas
1	max	var_global	0	0	tvector	$\Rightarrow (150, \text{treal})$
2	f1	funcion	0	—	tfuncion	$\Rightarrow (\text{tentero}, \text{tentero}) \rightarrow \text{tentero}$
3	p2	parámetro	1	-3	tentero	
4	p1	parámetro	1	-4	tentero	
5	v1	var_local	1	0	treal	
6	v2	var_local	1	2	treal	

1	max	var_global	0	0	tvector	$\Rightarrow (150, \text{treal})$
2	f1	funcion	0	—	tfuncion	$\Rightarrow (\text{tentero}, \text{tentero}) \rightarrow \text{tentero}$
3	main	funcion	0	—	tfuncion	$\Rightarrow (\text{tvacio}) \rightarrow \text{tentero}$
4	n	var_local	1	0	treal	



### Ejercicio 109

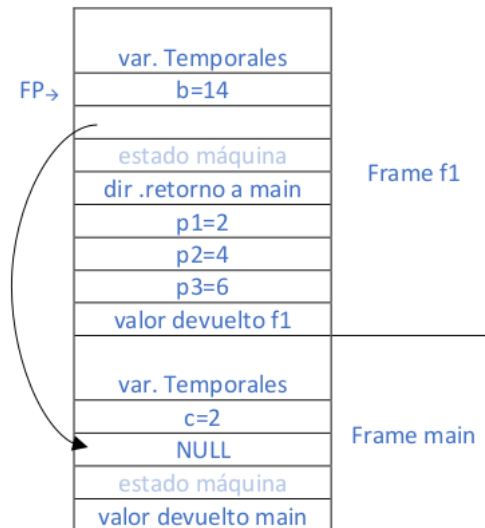
Respuesta a:

	nom		n	$\delta$	tipo	Subtablas
1	a	var_global	0	0	tentero	
2	f1	funcion	0	—	tfuncion	$\Rightarrow (\text{tentero}, \text{treal}, \text{tentero}) \rightarrow \text{tentero}$
3	p3	parámetro	1	-10	tentero	
4	p2	parámetro	1	-14	tentero	
5	p1	parámetro	1	-16	tentero	
6	b	var_local	1	0	treal	

Respuesta b:

$$\delta_{p1} = -16; \quad \delta_{p2} = -14;$$

Respuesta c:



### Ejercicio 110

Respuesta a:

nombre	categoría	nivel	desplazamiento	tipo
f1	función	0	—	tfunción((tentero,tentero)→treal
main	función	0	—	tfunción(tvacio→tvacio
a	variable	1	0	tentero
c	variable	1	2	treal
b	variable	1	6	tentero
a	variable	2	8	tentero
b	variable	2	10	tentero

Respuesta b:

Pto.A c (1, 2); a (2, 8); b (1, 6)

Pto.B c (1, 2); a (2, 8); b (2, 10)

Respuesta c:

\$	RA main	RA f1 (3, 2)	RA f1 (2, 2)
----	---------	--------------	--------------

### Ejercicio 111

nombre	categoría	nivel	desp.	tipo
G	función	0	—	tfunción((tentero,treal)→treal
b	parámetro	1	-12	treal
a	parámetro	1	-14	tentero
i	var-local	1	0	treal



nombre	categoría	nivel	desp.	tipo
G	función	0	—	tfunción((tentero,treal)→treal
x	var-glob	0	0	tregistro((c1, tentero, 0), (c2, treal, 2), (c3, tentero, 6))
mem	var-glob	0	8	tentero
main	función	0	—	tfunción(tvacio→tvacio
i	var-local	1	0	tentero
A	var-local	1	2	tvector(10, tentero)
k	var-local	1	22	treal

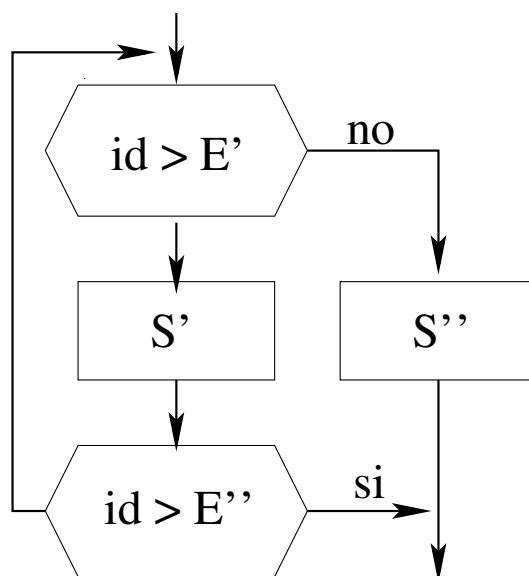
## 6. Generación de Código Intermedio

112. Dada la gramática, diseñad un ETDS que tenga en cuenta: 1) las comprobaciones semánticas de tipo; 2) la gestión de la TDS y la gestión de memoria para los objetos declarados; y 3) la generación de código intermedio para la instrucción propuesta.

$D \rightarrow T \text{ id}$	$T \rightarrow \text{union} \{ LC \}$	$C \rightarrow \text{id} . C$
$T \rightarrow \text{int} \mid \text{double}$	$LC \rightarrow LC ; T \text{ id}$	$C \rightarrow \text{id}$
$T \rightarrow \text{struct} \{ LC \}$	$LC \rightarrow T \text{ id}$	$S \rightarrow \text{id} . C = E$

113. Dada la instrucción *pseudowhile* cuya semántica viene definida por el diagrama de flujo que se adjunta. Diseñad un ETDS para la generación de código intermedio. Utilizad listas de referencias no satisfechas para el control de los saltos.

$S \rightarrow \text{pseudowhile id}, E' \text{ until } E'' \text{ do } S' \text{ then } S''$



114. \* Diseñad un ETDS que genere código intermedio para la siguiente instrucción repetitiva:

$I \rightarrow \text{while } E \text{ do } I \text{ R}$   
 $R \rightarrow \text{except } E' \text{ do } I' \mid \epsilon$

La instrucción **while** tiene el comportamiento habitual: mientras que  $E$  sea cierto ejecuta  $I$ . La parte **except** permite ejecutar una instrucción alternativa,  $I'$ , si su expresión,  $E'$ , es cierta. La parte **except** se evalúa ANTES de entrar en el cuerpo del bucle ( $I$ ) del **while** y después de analizar la expresión  $E$ .

Ejemplo:

```

a=0;
while a≤5 do
  { a++; write(a); }
except (a == 2) do { a=a+3; }
  
```

$\Rightarrow$  1 2 6

115. La siguiente gramática define una nueva instrucción que permite ejecutar (solo) las instrucciones de un determinado bloque de una lista de bloques. El número de bloque a ejecutar depende del valor entero de la expresión  $E$ : Si  $E$  vale 1 se ejecutará solo el primer bloque, si vale 2 el solo segundo, y así sucesivamente. Construid un ETDS que genere código intermedio para esta nueva instrucción:

$$\begin{array}{lll}
I & \rightarrow & \text{run } E \text{ in } LB \\
LB & \rightarrow & \{ LI \} LB \quad | \quad \epsilon \\
LI & \rightarrow & I ; LI \quad | \quad \epsilon
\end{array}$$

116. \* Dada la siguiente gramática, diseñad un ETDS que genere código intermedio. La instrucción **yacase** (*yet another case*) es similar a la del PASCAL: si la **cte** coincide con la expresión **E** debe ejecutar la secuencia de instrucciones **I** asociada, y terminar la búsqueda en la lista de ítems. La instrucción **exit** supone la salida inmediata de la instrucción **yacase**.

$$\begin{array}{ll}
I \rightarrow \text{yacase } E \text{ of } L \text{ default } I \text{ end} & L \rightarrow L ; \text{cte} : I \\
I \rightarrow I ; I & L \rightarrow \text{cte} : I \\
I \rightarrow \text{exit} & L \rightarrow \text{else } I
\end{array}$$

117. Diseñad un ETDS que genere código intermedio para el siguiente fragmento de una gramática:

$$\begin{array}{ll}
I \rightarrow \text{dependon } E1 \text{ execute } E2 \text{ times } \{ B \\
B \rightarrow \text{num} : LI ; B \quad | \quad \} \\
LI \rightarrow I ; LI \quad | \quad \epsilon
\end{array}$$

Donde la instrucción *dependon* ejecuta un bucle el número de veces indicado por *E2*. En cada iteración del bucle se ejecutarán todos los bloques de instrucciones precedidos del número cuyo valor coincide con el de la expresión *E1*. *E1* y *E2* no se modifican en el bucle.

**Ejemplo** La salida de este ejemplo será: **e e e**

```

dependon 5 execute 3 times {
    2: print(b);
    5: print(e);
    3: print(c); }

```

118. \* Diseñad un ETDS que genere código intermedio para el siguiente fragmento de una gramática:

$$S \rightarrow \text{repeat-if } E \text{ then } S \text{ else } S \text{ until } E$$

**repeat-if** es una instrucción repetitiva en la que, dependiendo del valor de la expresión  $E^1$ , se ejecutará  $S^1$  en caso de que sea TRUE y  $S^2$  en caso de FALSE. Este proceso se repetirá hasta que la expresión  $E^2$  sea TRUE.

119. Diseñad un ETDS que genere código intermedio para el siguiente fragmento de una gramática:

$$\begin{array}{ll}
I \rightarrow \text{for id in id do } I \text{ step } E \\
I \rightarrow \text{next} \quad | \quad \text{id} := E \quad | \quad I ; I
\end{array}$$

donde `id-1` es una variable de tipo simple e `id-2` de tipo array.

La instrucción `for` es un bucle que en cada iteración asigna a la variable `id-1` un elemento del array `id-2`: desde el 0 (en la primera iteración) hasta el límite superior del array (en la última). La expresión `E` del `for` indica cuál será el próximo elemento del array a asignar al `id-1` y debe ser  $\geq 1$  (p.ej. un `step 3` irá asignando a `id-1` los elementos 0, 3, 6 ... del array).

Cuando en el cuerpo del bucle se ejecutó la instrucción `next`, se abandonará inmediatamente la iteración en curso y se pasará al inicio de la siguiente.

120. Diseñad un ETDS que genere código intermedio para el bucle representado por la siguiente gramática:

$$I \rightarrow \text{repeat if } E_1 \text{ then } I_1 \text{ else } I_2 \text{ until } E_2 \text{ updating } I_3$$

Esta instrucción debe repetir el contenido del bucle hasta que  $E_2 = \text{true}$ , en cuyo caso se debe salir del mismo. En el interior del bucle, si  $E_1 = \text{true}$  entonces se ejecutará la secuencia de instrucciones  $I_1$  sino la  $I_2$ . Si  $E_2 = \text{false}$ , y antes de repetir el bucle, se debe ejecutar la secuencia de instrucciones  $I_3$ .

121. \* Diseñad un ETDS que realice las acciones semánticas necesarias para la comprobación de tipos y la generación de código intermedio para la siguiente gramática:

$$\begin{aligned} I &\rightarrow \text{select } \{ L \} \\ L &\rightarrow L ; E : I \mid E : I \end{aligned}$$

Donde  $I$  representa una secuencia de instrucciones y  $E$  una expresión lógica. La operación `select` debe evaluar la secuencia de expresiones lógicas en el orden que ocurran. Si alguna de ellas toma el valor `verdad` entonces se debe ejecutar la secuencia de instrucciones  $I$  que la acompaña y finalizar la evaluación de la instrucción `select`.

## 6.1. Ejercicios resueltos

### Ejercicio 116

$S \Rightarrow \text{yacase } E$	$\underline{\text{Si}} (E.t \neq \text{tentero}) \text{ MenError}(.);$
$\quad \text{of } L \text{ default } S$	$L.\text{pos} = E.\text{pos};$
	$\text{CompletaLans}(L.\text{fin}, \Omega);$
	$\text{CompletaLans}(\text{FusionaLans}(L.b, S^1.b), \Omega); \quad S.b = \text{nil};$
$\Rightarrow S ; S$	$S.b = \text{FusionaLans}(S^1.b, S^2.b);$
$\Rightarrow \text{exit}$	$S.b = \text{CreaLans}(\Omega); \quad \text{Emite}(\text{goto } \otimes);$
$L \Rightarrow$	$L^1.\text{pos} = L.\text{pos};$
$\quad L ; \text{cte}$	$\underline{\text{Si}} (\text{cte}.t \neq \text{tentero}) \text{ MenError}(.);$
	$L.\text{sal} = \text{CreaLans}(\Omega); \quad \text{Emite}(\text{if } L.\text{pos} \neq \text{cte.num goto } \otimes);$
$\quad : S$	$L.\text{fin} = \text{CreaLans}(\Omega); \quad \text{Emite}(\text{goto } \otimes);$
	$\text{CompletaLans}(L.\text{sal}, \Omega);$
	$L.\text{fin} = \text{FusionaLans}(L.\text{fin}, L^1.\text{fin});$
	$L.b = \text{FusionaLans}(L^1.b, S.b);$
$\Rightarrow L ; \text{cte}$	$\underline{\text{Si}} (\text{cte}.t \neq \text{tentero}) \text{ MenError}(.);$
	$L.\text{sal} = \text{CreaLans}(\Omega); \quad \text{Emite}(\text{if } L.\text{pos} \neq \text{cte.num goto } \otimes);$
$\quad : S$	$L.\text{fin} = \text{CreaLans}(\Omega); \quad \text{Emite}(\text{goto } \otimes);$
	$\text{CompletaLans}(L.\text{sal}, \Omega); \quad L.b = S.b;$

La regla (  $L \Rightarrow \text{else } S$  ) es redundante con la opción (  $\text{default } S$  ) y no hace falta hacerla.

### Ejercicio 118

$S \Rightarrow \text{repeat-if}$	$S.l0 = \Omega;$
$\quad ( E )$	$\underline{\text{Si}} (E^1.t \neq \text{tlógico}) \text{ MenError}(.);$
	$S.l1 = \text{CreaLans}(\Omega); \quad \text{Emite}(\text{if } E^1.\text{pos} = 0 \text{ goto } \otimes);$
$\quad \text{then } S$	$S.l2 = \text{CreaLans}(\Omega); \quad \text{Emite}(\text{goto } \otimes);$
	$\text{CompletaLans}(S.l1, \Omega);$
$\quad \text{else } S$	$\text{CompletaLans}(S.l2, \Omega);$
$\quad \text{until } E$	$\underline{\text{Si}} (E^2.t \neq \text{tlógico}) \text{ MenError}(.);$
	$\text{Emite}(\text{if } E^2.\text{pos} = 0 \text{ goto } S.l0);$

### Ejercicio 121

$I \Rightarrow \text{select } \{ L \}$	CompletaLans(L.fin, $\Omega$ );
$L \Rightarrow L ; E :$	<u>Si</u> (E.t $\neq$ tlógico) MenError(.); E.lf = CreaLans( $\Omega$ ); Emitte(if E.pos = 0 goto $\otimes$ ); I L.fin = FusionaLans(L <sup>1</sup> .fin, CreaLans( $\Omega$ )); Emitte(goto $\otimes$ ); CompletaLans(E.lf, $\Omega$ );
$L \Rightarrow E :$	<u>Si</u> (E.t $\neq$ tlógico) MenError(.); E.lf = CreaLans( $\Omega$ ); Emitte(if E.pos = 0 goto $\otimes$ ); I L.fin = CreaLans( $\Omega$ ); Emitte(goto $\otimes$ ); CompletaLans(E.lf, $\Omega$ );

### Ejercicio 114

$I \Rightarrow \text{while}$	L.ini = R.ini = $\Omega$ ;
E	I.v = CreaLans ( $\Omega$ ); Emitte (if E.pos = 1 goto $\otimes$ ); I.fin = CreaLans ( $\Omega$ ); Emitte (goto $\otimes$ ); R.f = $\Omega$ ;
I	Emitte (goto I.ini); CompletaLans (I.v, $\Omega$ );
R	CompletaLans (I.fin, $\Omega$ );
$R \Rightarrow E$	Emitte (if E.pos = 0 goto R.f);
I	Emitte (goto R.ini);;

## 7. Optimización de Código Intermedio

122. \* Dado el siguiente fragmento de un bloque básico, aplicad las optimizaciones locales mediante la construcción de su GDA y la posterior reconstrucción del código.

$t_1 = 2$	$t_5 = t_4 + 1$	$z = t_8$
$t_2 = y + t_1$	$t_6 = x - t_5$	$t_9 = 10$
$x = t_2$	$y = t_6$	$t_{10} = x - t_9$
$t_3 = 1$	$t_7 = 2$	$w = t_{10}$
$t_4 = t_3 * 9$	$t_8 = y + t_7$	

123. Dado el siguiente fragmento de código intermedio, indicad los bloques básicos que forman el bucle y sus familias de variable de inducción (con sus ternas asociadas). A continuación, aplicad los algoritmos de extracción de código invariante, reducción de intensidad y de eliminación de variables de inducción.

100 $j = 1$	105 $t_3 = t_2 + j$	110 $a[t_4] = j$
101 $i = 0$	106 $a[t_1] = t_3$	111 $i = i + 1$
102 <i>if</i> $i > N$ <i>goto</i> 108	107 <i>goto</i> 111	112 <i>if</i> $i \leq 100$ <i>goto</i> 102
103 $t_1 = i * 4$	108 $t_4 = i * 4$	113 ...
104 $t_2 = a[t_1]$	109 $j = 0$	

124. Dado el siguiente fragmento de código intermedio:

100 $i := 0$	109 $t_7 := t_6 + j$	118 $t_{14} := j * 20$
101 $j := 10$	110 $t_8 := t_7 * 4$	119 $t_{15} := t_{14} + i$
102 <i>if</i> $i > 5$ <i>goto</i> 113	111 $a[t_8] := t_5$	120 $t_{16} := t_{15} * 4$
103 $t_1 := i * 20$	112 <i>goto</i> 122	121 $a[t_{16}] := t_{13}$
104 $t_2 := t_1 + j$	113 $t_9 := j * 20$	122 $t_{17} := 1$
105 $t_3 := t_2 * 4$	114 $t_{10} := t_9 + i$	123 $t_{18} := i + t_{17}$
106 $t_4 := a[t_3]$	115 $t_{11} := t_{10} * 4$	124 $i := t_{18}$
107 $t_5 := t_4 + j$	116 $t_{12} := a[t_{11}]$	125 <i>if</i> $i < 10$ <i>goto</i> 102
108 $t_6 := i * 20$	117 $t_{13} := t_{12} + j$	...

- a) Determinad los bloques básicos y el grafo de flujo. A partir de los GDAs, reconstruid el código aplicando las optimizaciones locales.
- b) Indicad los bloques básicos que forman el bucle y sus familias de variables de inducción (con sus ternas asociadas). A continuación, aplicad los algoritmos de extracción de código invariante, reducción de intensidad y de eliminación de variables de inducción.
125. Dado el siguiente fragmento de código intermedio de un bloque básico, aplicad las optimizaciones locales a partir de su GDA. A la salida del bloque solo estarán activas las variables:  $A$   $i$ .

```

(100)   $t_0 := 0$ 
(101)   $i := t_0$ 
(102)   $y := k$ 
(103)   $x := 0$ 
(104)   $t_1 := y + x$ 
(105)   $y := t_1$ 
(106)   $t_2 := i * 4$ 
(107)   $t_3 := t_2 + 2$ 
(108)   $t_4 := t_3 * y$ 
(109)   $t_5 := A[t_4]$ 
(110)   $x := x + t_4$ 
(111)   $i := y$ 
(112)  if  $i < x$  goto 200

```

126. Dado el siguiente fragmento de código intermedio:

```

(100)   $s := 0$ 
(101)   $m := 1$ 
(102)   $t_1 := m * 2$ 
(103)   $ini := size * 2$ 
(104)   $t_2 := t_1 - ini$ 
(105)  if  $x > y$  goto 108
(106)   $s := a[t_2]$ 
(107)  goto 109
(108)   $s := b[t_2]$ 
(109)   $tot := tot + s$ 
(110)   $m := m + 2$ 
(111)  if  $m < 100$  goto 102
(112)  print( $s$ )

```

- a) Determinad los bloques básicos que forman el/los bucle/s. Extrae el código invariante. Indicad las variables de inducción y sus ternas asociadas.
- b) Aplicad el algoritmo de reducción de intensidad.
- c) Aplicad el algoritmo de eliminación de variables de inducción.

127. \* Dado el siguiente fragmento de código intermedio de un bloque básico, aplicad las optimizaciones locales a partir de su GDA. A la salida del bloque solo estarán activas las variables: **a**, **b**, **x**, **y**, **z**.

(100) $t_1 := 10$	(106) $t_5 := c + x$	(112) $z := x + y$
(101) $t_2 := t_1 * 2$	(107) $x := t_5$	(113) $t_8 := 1$
(102) $c := t_2$	(108) $t_6 := d * y$	(114) $a := z + t_9$
(103) $t_3 := 5$	(109) $b := c + x$	(115) $t_9 := 1$
(104) $t_4 := t_3 + 2$	(110) $t_7 := d * y$	(116) $z := x * y$
(105) $d := t_4$	(111) $y := t_7$	(117) $a := b + 1$

128. \* Dado el siguiente fragmento de código intermedio:



(100) $k := 0$	(105) $t_4 := 5$	(110) $t_9 := t_3 + t_8$
(101) $m := 100$	(106) $t_5 := t_4 * 2$	(111) $A[t_2] := t_9$
(102) $t_1 := k * 2$	(107) $t_6 := t_1 + t_5$	(112) <i>if</i> $k > m$ <i>goto</i> 115
(103) $t_2 := t_1 * 2$	(108) $t_7 := t_6 * 2$	(113) $k := k + 1$
(104) $t_3 := A[t_2]$	(109) $t_8 := A[t_7]$	(114) <i>goto</i> 102

- Determinad los bloques básicos que forman el bucle. Extraed el código invariante e indicad las variables de inducción y sus ternas asociadas.
- Aplicad el algoritmo de reducción de intensidad.
- Aplicad el algoritmo de eliminación de variables de inducción.

129. Dado el siguiente fragmento de código intermedio de un bloque básico, aplicad las optimizaciones locales a partir de su GDA. A la salida del bloque, aparte de las variables  $i$ ,  $j$ ,  $A$ ,  $B$ , solo estará activa la variable:  $k$ .

(100) $t_1 = 7$	(103) $t_4 = A[t_3]$	(106) $t_7 = t_6$	(109) $t_{10} = t_7 * t_9$
(101) $t_2 = t_1 * 4$	(104) $t_5 = 7$	(107) $t_8 = 0$	(110) $t_{11} = B[t_{10}]$
(102) $t_3 = i * t_2$	(105) $t_6 = t_5 * 4$	(108) $t_9 = j + t_8$	(111) $k = t_4 + t_{11}$

130. Dado el siguiente fragmento de código intermedio:

(100) $i = 5$	(104) $t_2 = 10$	(108) $x = x + t_5$	(112) $t_7 = A[t_6]$
(101) $j = 20$	(105) $t_3 = t_2 * 2$	(109) $j = j - 2$	(113) $x = x + t_7$
(102) $x = 0$	(106) $t_4 = t_1 + t_3$	(110) <i>if</i> $i > 10$ <i>goto</i> 116	(114) $i = i + 2$
(103) $t_1 = j * 4$	(107) $t_5 = B[t_4]$	(111) $t_6 = i * 2$	(115) <i>goto</i> 103

- Determinad los bloques básicos que forman el/los bucle/s. Extraed el código invariante. Indicad las variables de inducción y sus ternas asociadas.
- Aplicad el algoritmo de reducción de intensidad.
- Aplicad el algoritmo de eliminación de variables de inducción.

131. \* Dado el siguiente fragmento de código intermedio, y sabiendo que a la salida del bucle solo estará activa la variable:  $a$ ,

(100) $k := 0$	(106) $t_4 := a[t_3]$	(112) $t_8 := t_7 * 4$
(101) $L := 10$	(107) $t_5 := t_4 + 10$	(113) $t_9 := A[t_8]$
(102) <i>if</i> $k > 5$ <i>goto</i> 110	(108) $a[t_3] := t_5$	(114) $t_{10} := t_9 + 10$
(103) $t_1 := K * 20$	(109) <i>goto</i> 116	(115) $a[t_8] := t_{10}$
(104) $t_2 := t_1 + L$	(110) $t_6 := L * 20$	(116) $k := k + 1$
(105) $t_3 := t_2 * 4$	(111) $t_7 := t_6 + k$	(117) <i>if</i> $k < 10$ <i>goto</i> 102

- Determinad los bloques básicos que forman el bucle. Extraed el código invariante e indicad las variables de inducción y sus ternas asociadas.
- Aplicad el algoritmo de reducción de intensidad.
- Aplicad el algoritmo de eliminación de variables de inducción.

132. Dado el siguiente fragmento de código intermedio de un bloque básico, aplicad las optimizaciones locales a partir de su GDA. A la salida del bloque, aparte de las variables  $i$ ,  $j$ ,  $A$ ,  $B$ , solo estará activa la variable:  $k$ .

(100) $t_1 = 7$	(103) $t_4 = A[t_3]$	(106) $t_7 = t_6$	(109) $t_{10} = t_7 * t_9$
(101) $t_2 = t_1 * 4$	(104) $t_5 = 7$	(107) $t_8 = 0$	(110) $t_{11} = B[t_{10}]$
(102) $t_3 = i * t_2$	(105) $t_6 = t_5 * 4$	(108) $t_9 = j + t_8$	(111) $k = t_4 + t_{11}$

133. Dado el siguiente fragmento de código intermedio:

(100) $i = 5$	(104) $t_2 = 10$	(108) $x = x + t_5$	(112) $t_7 = A[t_6]$
(101) $j = 20$	(105) $t_3 = t_2 * 2$	(109) $j = j - 2$	(113) $x = x + t_7$
(102) $x = 0$	(106) $t_4 = t_1 + t_3$	(110) <i>if</i> $i > 10$ <i>goto</i> 116	(114) $i = i + 2$
(103) $t_1 = j * 4$	(107) $t_5 = B[t_4]$	(111) $t_6 = i * 2$	(115) <i>goto</i> 103

- a) Determinad los bloques básicos que forman el/los bucle/s. Extraed el código invariante. Indicad las variables de inducción y sus ternas asociadas.
- b) Aplicad el algoritmo de reducción de intensidad.
- c) Aplicad el algoritmo de eliminación de variables de inducción.
134. \* Dado el siguiente fragmento de un bloque básico, aplicad las optimizaciones locales mediante la construcción de su GDA y la posterior reconstrucción del código. Considerad activas a la salida del bloque las variables:  $x$ ,  $y$ ,  $k$

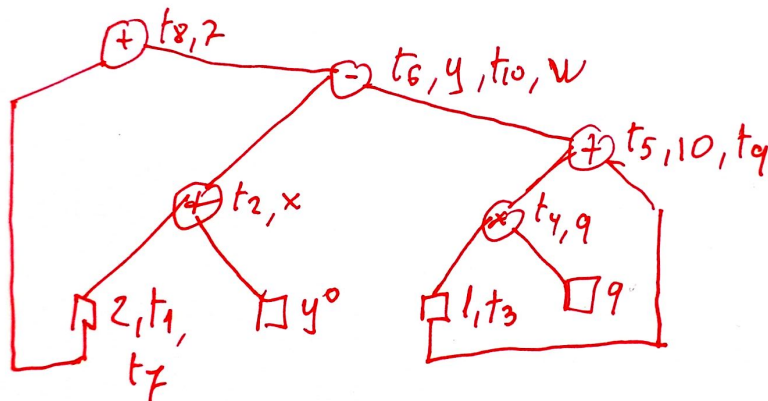
$t_2 = 3$	$y = t_6$	$t_{11} = 7$	$t_{15} = 40$
$t_3 = y + t_2$	$t_7 = 1$	$t_{12} = t_{11} + x$	$t_{16} = t_{15} + k$
$t_4 = 4$	$t_8 = t_7 * x$	$k = t_{12}$	$x = t_{16}$
$t_5 = t_4 * 10$	$t_9 = 0$	$t_{13} = 7$	
$t_6 = t_5 + k$	$t_{10} = t_9 + t_8$	$t_{14} = t_{13} + x$	

135. Dado el siguiente fragmento de código intermedio, indicad los bloques básicos que forman el bucle y sus variables de inducción (con sus ternas asociadas). A continuación, aplicad las optimizaciones globales de extracción de código invariante, reducción de intensidad y de eliminación de variables de inducción.

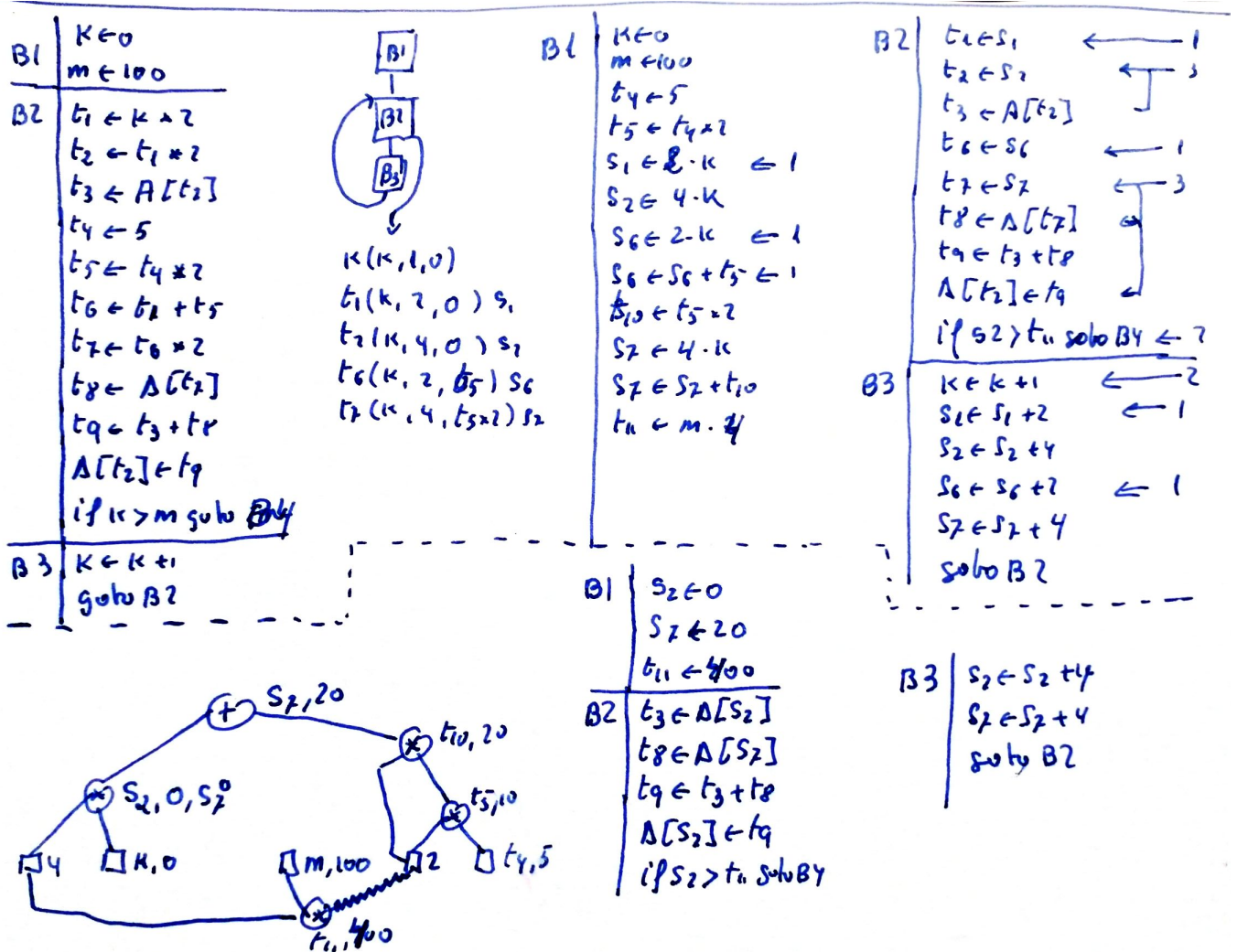
100 $j = 10$	104 $t_5 = a[t_4]$	108 $x = t_8 + t_5$	112 $j = j + 1$
101 $i = 1$	105 $t_6 = 10 + i$	109 <i>if</i> $x \geq M$ <i>goto</i> 112	113 <i>if</i> $i > 20$ <i>goto</i> 115
102 $t_3 = 70 + j$	106 $t_7 = t_6 * 2$	110 $i = i + 1$	114 <i>if</i> $j \leq 10$ <i>goto</i> 102
103 $t_4 = t_3 * 2$	107 $t_8 = b[t_7]$	111 <i>goto</i> 113	115 ...

### 7.1. Ejercicios resueltos

## Ejercicio 122

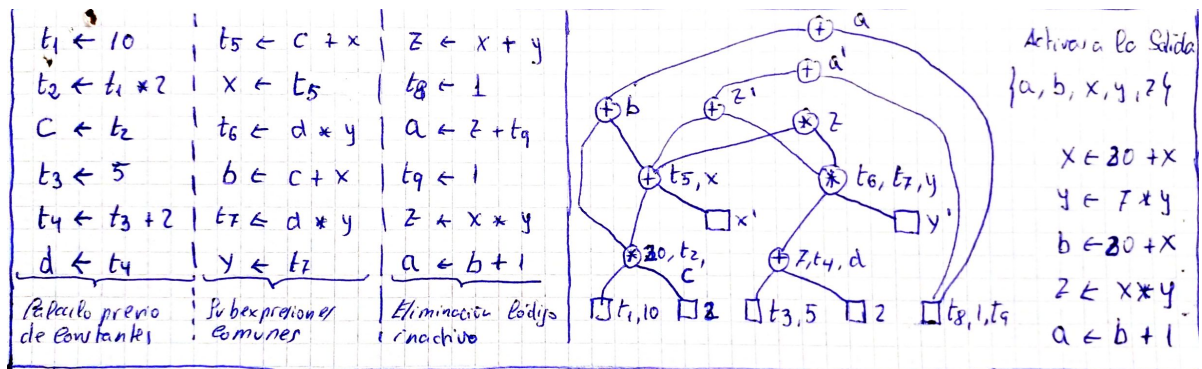

$$\begin{aligned}x &\leftarrow 2 + y \\w &\leftarrow x \div 10 \\z &\leftarrow w + 2\end{aligned}$$

## Ejercicio 128

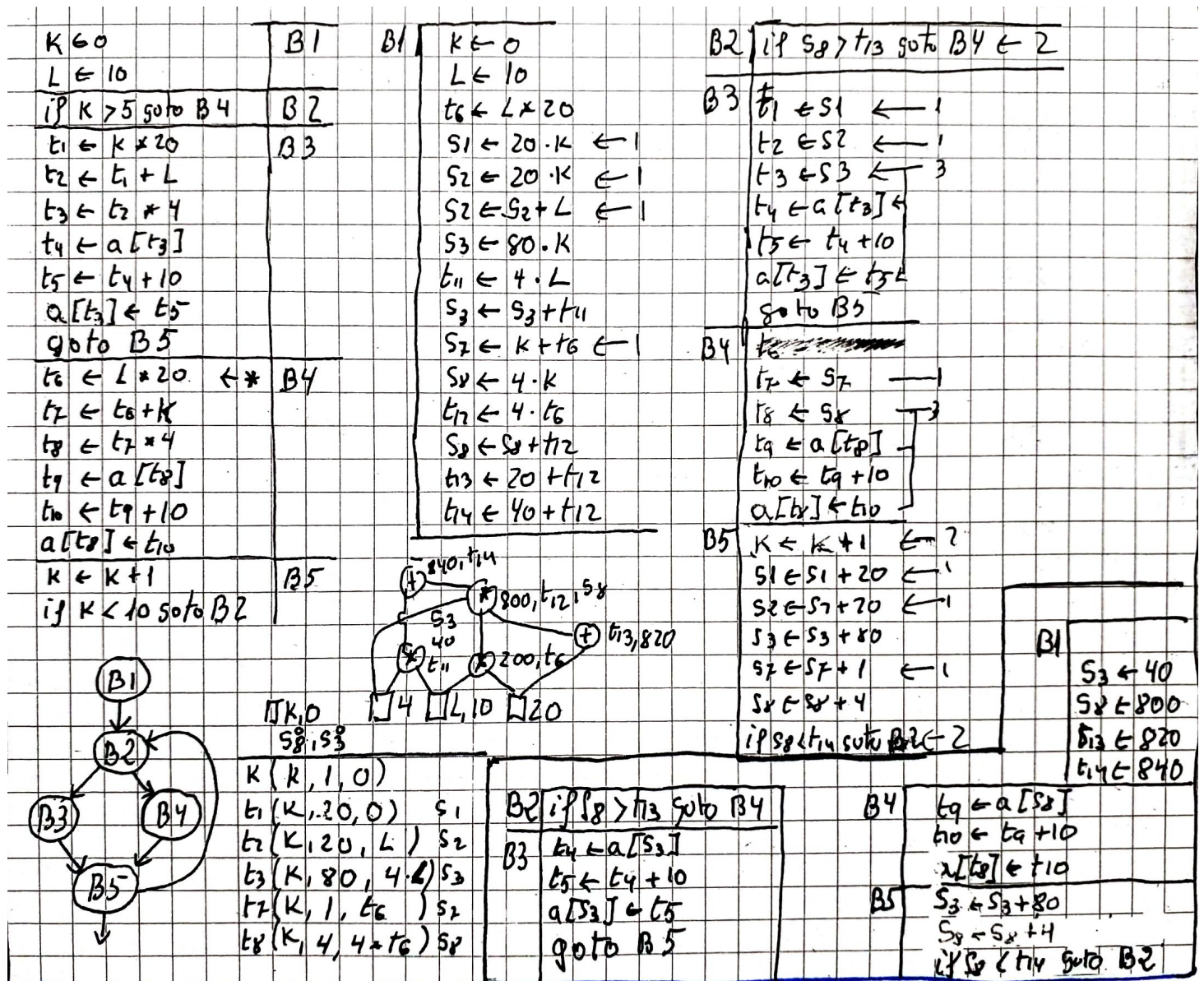




## Ejercicio 127

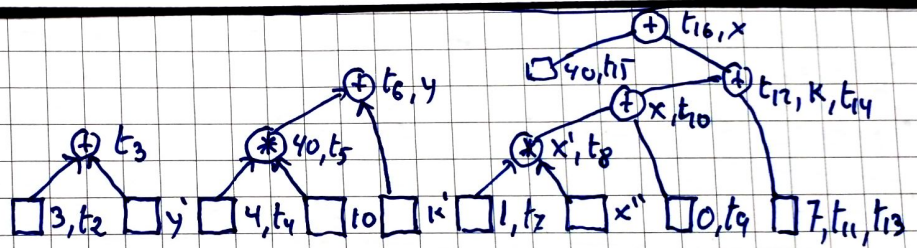


## Ejercicio 131



# Ejercicio 134

$t_2 \leftarrow 3$	
$t_3 \leftarrow t_2 + y$	
$t_4 \leftarrow 4$	
$t_5 \leftarrow t_4 \times 10$	
$t_6 \leftarrow t_5 + k$	
$y \leftarrow t_6$	
$t_7 \leftarrow 1$	
$t_8 \leftarrow t_7 \times x$	
$t_9 \leftarrow 0$	
$t_{10} \leftarrow t_9 + t_8$	
$t_{11} \leftarrow 7$	
$t_{12} \leftarrow t_{11} + x$	
$k \leftarrow t_{12}$	
$t_{13} \leftarrow 7$	
$t_{14} \leftarrow t_{13} + x$	
$t_{15} \leftarrow 40$	
$t_{16} \leftarrow t_{15} + k$	
$x \leftarrow t_{16}$	



$$y \leftarrow 40 + k$$

$$k \leftarrow x + 7$$

$$x \leftarrow 40 + k$$