Recopilación de ejercicios sobre esquemas de traducción en exámenes de *Compiladores e intérpretes*

Mayo de 2006

Ejercicio 1 — Valor: 2,00 puntos

Considera un lenguaje de listas con las siguientes características:

- Cada lista empieza con un componente léxico **ini** y termina con un componente léxico **fin**.
- Cada lista está dividida en dos partes separadas por un componente léxico **sep**: una cabecera de lista y una secuencia de elementos.
- La cabecera de una lista consiste en un identificador cuya utilización se va a permitir dentro de la secuencia de elementos.
- En esta secuencia pueden aparecer tanto identificadores como listas anidadas.

Considera, además, la gramática

$$\begin{aligned}
&\langle \text{UnaLista} \rangle \rightarrow \textbf{ini}\ \langle \text{Cabecera} \rangle\ \textbf{sep}\ \langle \text{Secuencia} \rangle\ \textbf{fin} \\
&\langle \text{Cabecera} \rangle \rightarrow \textbf{id} \\
&\langle \text{Secuencia} \rangle \rightarrow \langle \text{Elemento} \rangle\ \langle \text{Secuencia} \rangle \mid \lambda \\
&\langle \text{Elemento} \rangle \rightarrow \textbf{id} \mid \langle \text{UnaLista} \rangle
\end{aligned}$$

y que los componentes léxicos de la categoría **id** (es decir, los identificadores) tienen un atributo *lex* donde se guarda el correspondiente lexema.

Añade a la gramática anterior las acciones pertinentes para que se llame a una función `error` en caso de que, como elemento de una lista, se utilice un identificador no permitido, esto es, uno que no haya aparecido en la cabecera de ninguna de las listas que lo contienen.

Ejercicio 2 — Valor: 2,50 puntos

Se desea modelar un lenguaje de matrices numéricas en el que, por ejemplo, la matriz de dos filas y tres columnas

$$\begin{pmatrix} 1 & -3 & 0 \\ -1 & 0 & -2 \end{pmatrix}$$

quedaría representada como sigue:

```
( 1 -3 0 ; -1 0 -2 ; )
```

Además, la especificación léxica está dada:

Categoría léxica	Expresión regular	Emitir u omitir	Atributos
blanco	`[ \t\n]+`	Omitir	—
número	`-?[0-9]+`	Emitir	*valor*
abre	`\(`	Emitir	—
cierra	`\)`	Emitir	—
finfila	`;`	Emitir	—

Así, en el ejemplo anterior, lo que vería el analizador sintáctico sería la siguiente secuencia de categorías:

abre número número número finfila número número número finfila cierra

Mediante el modelado sintáctico, se pretende reflejar lo siguiente:

- Una cadena correcta del lenguaje es aquella que representa a una única matriz.
- Una matriz empieza con un componente léxico **abre** y acaba con **cierra**.

- Una matriz tiene una o más filas.
- Cada fila consta de uno o más componentes léxicos **número** y acaba con **finfila**.

En primer lugar, debes modelar, mediante una gramática incontextual con símbolo inicial ⟨Matriz⟩ y sin partes derechas regulares, el nivel sintáctico de este lenguaje de matrices.

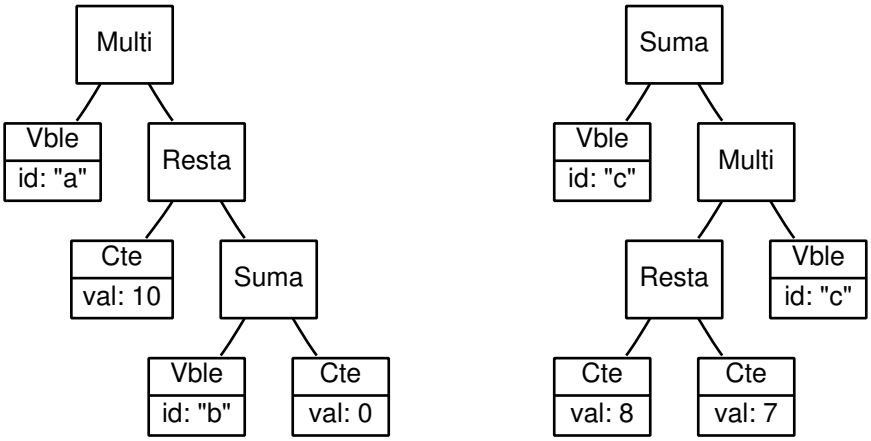
Además, como se desea controlar que todas las filas de la matriz tengan el mismo número de elementos, debes añadir a la gramática anterior las acciones pertinentes para que se llame a una función `error` si alguna fila de la matriz no tiene el mismo número de elementos que la primera. Ten muy en cuenta que en este esquema de traducción no se te permite hacer uso de tablas de símbolos ni ningún otro tipo de objeto global: ha de bastarte con utilizar atributos asociados a los no terminales de la gramática.

Ejercicio 3 — Valor: 3,50 puntos

Se desea modelar un lenguaje de expresiones, al que llamaremos *LEDi3*, con las siguientes características:

- Las únicas operaciones permitidas serán suma, resta y multiplicación, las tres con notación binaria infija.
- Todas tendrán el mismo nivel de precedencia y la asociatividad será por la derecha.
- Suma y resta se representarán mediante los símbolos habituales; la multiplicación, mediante una equis mayúscula o minúscula.
- Podrán utilizarse paréntesis de la forma habitual y, además, también podrán utilizarse corchetes para parentizar expresiones.
- Sin embargo, no se podrá utilizar un corchete para cerrar un paréntesis, ni un paréntesis para cerrar un corchete.
- Los números serán todos enteros y se representarán mediante literales octales (es decir, en base ocho) sin signo.
- En las expresiones también podrán intervenir identificadores de variable, donde cada uno de estos identificadores será una letra minúscula distinta de equis.
- Una cadena válida del lenguaje no podrá contener ningún blanco y consistirá en una expresión doblemente parentizada por corchetes.

Así, las cadenas `[[aX12-b+0]]` y `[[c+[10-7]x(c)]]` serían, en *LEDi3*, representaciones válidas de las expresiones cuyos árboles de sintaxis abstracta se muestran a continuación:



Asume para *LEDi3* la siguiente especificación léxica, donde los atributos *lex* se utilizan para almacenar los correspondientes lexemas y *val* sirve para guardar el valor numérico extraído del correspondiente literal octal:

Categoría léxica	Expresión regular	Emitir u omitir	Atributos
opad	`[-+]`	Emitir	*lex*
opmul	`[Xx]`	Emitir	—
octal	`[0-7]+`	Emitir	*val*
ident	`[a-wyz]`	Emitir	*lex*
ap	`\(`	Emitir	—
cp	`\)`	Emitir	—
ac	`\[`	Emitir	—
cc	`\]`	Emitir	—

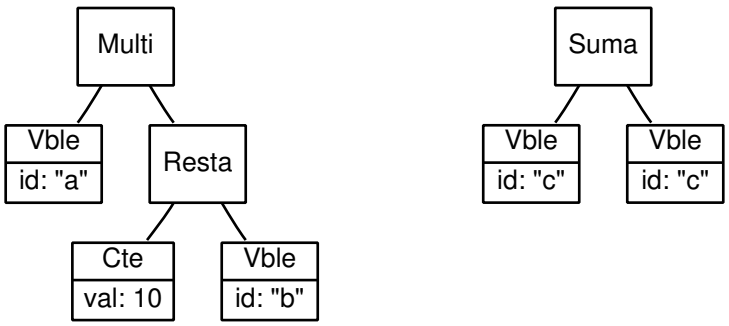
Además, considera que el nivel sintáctico de *LEDi3* viene modelado por la siguiente gramática LL(1):

$$\langle \text{Cad} \rangle \rightarrow \textbf{ac ac } \langle \text{Exp} \rangle \textbf{ cc cc}$$
$$\langle \text{Exp} \rangle \rightarrow \langle \text{Sub} \rangle \ \langle \text{Fin} \rangle$$
$$\langle \text{Fin} \rangle \rightarrow \textbf{opad } \langle \text{Exp} \rangle \mid \textbf{opmul } \langle \text{Exp} \rangle \mid \lambda$$
$$\langle \text{Sub} \rangle \rightarrow \textbf{octal} \mid \textbf{ident} \mid \textbf{ap } \langle \text{Exp} \rangle \textbf{ cp} \mid \textbf{ac } \langle \text{Exp} \rangle \textbf{ cc}$$

Se trata de que añadas a la gramática anterior las acciones pertinentes para sintetizar en un atributo *ast* de su símbolo inicial ⟨Cad⟩ un árbol de sintaxis abstracta adecuadamente simplificado, teniendo en cuenta todas las observaciones siguientes:

- En el esquema de traducción que se te pide no se te permite hacer uso de ningún tipo de objeto global: ha de bastarte con utilizar atributos asociados a los símbolos de la gramática.
- Para, cuando proceda, poder referirte de forma correcta a los atributos de los símbolos terminales, ten muy presente la especificación léxica dada para *LEDi3*.
- Para crear los diferentes tipos de nodos del árbol de sintaxis abstracta, dispones de las siguientes funciones:
  - `HazSuma(izq,der)`: Devuelve un árbol que representa a una suma y que se construye a partir de los árboles `izq` y `der` que representan, respectivamente, a los operandos izquierdo y derecho de esa suma.
  - `HazResta(izq,der)`: Esta función, de perfil análogo al de `HazSuma`, devuelve un árbol que representa a una resta.
  - `HazMulti(izq,der)`: Esta función, de perfil análogo al de las dos anteriores, devuelve un árbol que representa a una multiplicación.
  - `HazCte(val)`: Esta función devuelve un nodo hoja que representa al valor constante que se le pasa como parámetro.
  - `HazVble(id)`: Esta función devuelve un nodo hoja que representa a la variable cuyo identificador se le pasa como parámetro.
- Tu esquema de traducción final ha de crear un árbol simplificado de la expresión original, para lo cual deberá respetar las siguientes restricciones:
  - Nunca se creará un árbol que represente a una operación entre valores constantes.
  - Nunca se creará un árbol que represente a una suma con un sumando nulo.
  - Nunca se creará un árbol que represente a una resta con su sustraendo nulo.
  - Nunca se creará un árbol que represente a una multiplicación con un factor que valga cero o uno.

Así, los árboles creados para las cadenas `[[aX12-b+0]]` y `[[c+[10-7]x(c)]]` deberían ser éstos:



En el primer caso, se ha aplicado $\boxed{b+0=b}$; en el segundo, $\boxed{8-7=1}$ y $\boxed{1\times c=c}$.
Puede resultarte útil abordar el ejercicio en dos fases:

A. Primero, escribe un esquema de traducción que se preocupe únicamente de sintetizar en ⟨Cad⟩.*ast* una representación semántica correcta de la entrada, sin intentar aplicar ninguna simplificación al correspondiente árbol de sintaxis abstracta.

B. Después, modifica el esquema anterior para que los árboles construidos estén adecuadamente simplificados, respetando las restricciones anteriormente enunciadas.

Ejercicio 4 — Valor: 2,00 puntos

Sea L_1 el lenguaje de todas las cadenas formadas por cero o más letras minúsculas y que no tienen tres bes seguidas en su interior. Así, por ejemplo, las siguientes cadenas pertenecerían a L_1: `xyz`, `bb`, `bebebe`, `baobab`, λ. Pero no estas otras: `Gato`, `abbbba`, `xxxbbb`, `salu2`...

Asumiremos una especificación léxica en la que se han definido dos únicas categorías, ambas emitidas por el analizador léxico y sin atributos: **be**, para la letra be minúscula, y **otra**, para cualquier letra minúscula distinta de la be; asumiremos, además, la gramática siguiente:

$$\langle \mathsf{C} \rangle \rightarrow \langle \mathsf{M} \rangle \ (\ \langle \mathsf{M} \rangle \ )^* \mid \lambda$$
$$\langle \mathsf{M} \rangle \rightarrow \textbf{be} \mid \textbf{otra}$$

Debes añadir acciones semánticas a la gramática anterior de modo que el esquema de traducción resultante, ante cadenas aceptadas por la gramática y que no pertenezcan a L_1, reaccione llamando a una función `trata_error` que ya se encargará de dar un mensaje adecuado y abortar el análisis. En tu esquema de traducción no puedes hacer uso de ningún tipo de objeto global: debes utilizar atributos y, si lo crees conveniente, también variables locales.

Ejercicio 5 — Valor: 2,50 puntos

Un cierto lenguaje de cadenas de caracteres se define mediante dos niveles, léxico y sintáctico, dados por la especificación

Categoría léxica	Expresión regular	Emitir u omitir	Atributos
may	`[A-Z]`	Emitir	*lexema*
min	`[a-z]`	Emitir	*lexema*
dig	`[0-9]`	Emitir	*valor*

y la gramática siguiente:

$$\langle \mathsf{X} \rangle \rightarrow \langle \mathsf{X} \rangle \ \textbf{min} \ \langle \mathsf{Y} \rangle \ \langle \mathsf{Z} \rangle$$
$$\langle \mathsf{X} \rangle \rightarrow \textbf{may}$$
$$\langle \mathsf{Y} \rangle \rightarrow \langle \mathsf{Y} \rangle \ \textbf{dig}$$
$$\langle \mathsf{Y} \rangle \rightarrow \lambda$$
$$\langle \mathsf{Z} \rangle \rightarrow \textbf{dig} \ \langle \mathsf{Z} \rangle$$
$$\langle \mathsf{Z} \rangle \rightarrow \textbf{may}$$
$$\langle \mathsf{Z} \rangle \rightarrow \lambda$$

Debes añadir acciones semánticas a la gramática anterior de modo que el esquema de traducción resultante sintetice en un atributo lógico $\langle \mathsf{X} \rangle$.*ados* un valor que indique si la cadena de caracteres analizada acaba en el dígito dos o no. Así, por ejemplo, debería sintetizarse un valor lógico *verdadero* para casos como `Bb722` o `Dd123x2`; sin embargo, para casos como `A`, `XaFg77` o `Ms2Fx`, el resultado del cálculo debería ser *falso*. Además, has de observar las siguientes restricciones a la hora de construir el esquema de traducción:

- Sólo se permiten acciones semánticas en el extremo derecho de la parte derecha de cada producción, lo que obliga a que todos los atributos utilizados deban ser sintetizados.
- Sólo se permiten atributos de tipo entero o lógico.
- No puede utilizarse ningún objeto global.

Ejercicio 6 — Valor: 2,50 puntos

Sea un analizador de cadenas de caracteres definido mediante la especificación léxica

Categoría léxica	Expresión regular	Emitir u omitir	Atributos
más	`\+`	Emitir	—
ent	`0|-?[1-9][0-9]*`	Emitir	*valor*
acor	`\[`	Emitir	—
ccor	`\]`	Emitir	—
vale	`=`	Emitir	—
equis	`x`	Emitir	—

y el siguiente esquema de traducción:

$$\langle \mathsf{S} \rangle \rightarrow \langle \mathsf{A} \rangle \ \{\ \langle \mathsf{P} \rangle.x := \langle \mathsf{A} \rangle.x\ \}\ \langle \mathsf{P} \rangle\ \{\ \langle \mathsf{S} \rangle.v := \langle \mathsf{P} \rangle.v\ \}$$
$$\langle \mathsf{A} \rangle \rightarrow \textbf{acor equis vale ent ccor}\ \{\ \langle \mathsf{A} \rangle.x := \textbf{ent}.\mathit{valor}\ \}$$
$$\langle \mathsf{P} \rangle \rightarrow \{\ \langle \mathsf{M} \rangle.x := \langle \mathsf{P} \rangle.x\ \}\ \langle \mathsf{M} \rangle\ \textbf{más}\ \{\ \langle \mathsf{P} \rangle_1.x := \langle \mathsf{P} \rangle.x\ \}\ \langle \mathsf{P} \rangle_1\ \{\ \langle \mathsf{P} \rangle.v := \langle \mathsf{M} \rangle.v + \langle \mathsf{P} \rangle_1.v\ \}$$
$$\langle \mathsf{P} \rangle \rightarrow \lambda\ \{\ \langle \mathsf{P} \rangle.v := 0\ \}$$
$$\langle \mathsf{M} \rangle \rightarrow \langle \mathsf{E} \rangle_1\ \textbf{equis}\ \langle \mathsf{E} \rangle_2\ \{\ \langle \mathsf{M} \rangle.v := \langle \mathsf{E} \rangle_1.v \times (\langle \mathsf{M} \rangle.x)^{\langle \mathsf{E} \rangle_2.v}\ \}$$
$$\langle \mathsf{E} \rangle \rightarrow \textbf{ent}\ \{\ \langle \mathsf{E} \rangle.v := \textbf{ent}.\mathit{valor}\ \}$$
$$\langle \mathsf{E} \rangle \rightarrow \lambda\ \{\ \langle \mathsf{E} \rangle.v := 1\ \}$$

Supón como entrada la cadena de diecisiete caracteres `[x=2]x2+10x4+-3x+` y responde a las siguientes cuestiones sobre el comportamiento del analizador:

A. Di cuál sería la secuencia de *componentes* emitida por el nivel léxico del analizador.

B. Di cuál sería la correspondiente secuencia de *categorías léxicas*.

C. Dibuja el árbol de análisis que el nivel sintáctico del analizador asociaría a la anterior secuencia de categorías.

D. Vuelve a dibujar el árbol anterior, pero esta vez añade a cada nodo del árbol sus correspondientes atributos:

- A los nodos de terminales, si procede, los atributos calculados por el análisis léxico.
- A los nodos de símbolos no terminales, los atributos calculados por el análisis semántico.

Ejercicio 7 — Valor: 1,50 puntos

Sea un lenguaje definido mediante dos niveles, léxico y sintáctico, dados por la especificación

Categoría léxica	Expresión regular	Emitir u omitir	Atributos
a	`[Aa]`	Emitir	—
be	`[Bb]`	Emitir	—
ce	`[Cc]`	Emitir	—
apar	`\(`	Emitir	—
cpar	`\)`	Emitir	—
lit	`[0-9]+`	Emitir	*valor*

y la gramática siguiente:

$$\langle \mathsf{S} \rangle \rightarrow \langle \mathsf{L} \rangle\ \langle \mathsf{N} \rangle$$
$$\langle \mathsf{L} \rangle \rightarrow \textbf{a a a}\ \langle \mathsf{L} \rangle \mid \textbf{be a a}\ \langle \mathsf{L} \rangle \mid \textbf{ce a}$$
$$\langle \mathsf{N} \rangle \rightarrow \textbf{apar lit cpar}$$

Debes añadir acciones semánticas a la gramática anterior de modo que el esquema de traducción resultante, mientras se lleva a cabo un análisis LL(1) de la entrada, pueda hacer los cálculos necesarios para acabar obteniendo en un atributo $\langle \mathsf{N} \rangle.\mathit{ok}$ un valor lógico que indique si el valor del literal entero entre paréntesis coincide exactamente con el número de aes de la cadena de caracteres analizada. Así, por ejemplo, debería sintetizarse un valor lógico *verdadero* para casos como `aAAca(4)` o `BaaBaaCA(5)`; sin embargo, para casos como `ca(007)`, `aAAca(3)` o `baaAAAbaaCA(11)`, el resultado del cálculo debería ser *falso*. Además, has de observar las siguientes restricciones a la hora de construir el esquema de traducción: sólo se permiten atributos de tipo entero o lógico y no puede utilizarse ningún objeto global.

Ejercicio 8 — Valor: 3,00 puntos

Considera el conjunto C formado por las cadenas de dígitos que presentan la siguiente estructura: una secuencia de uno o más unos, seguida de una secuencia de uno o más doses y, para finalizar la cadena, una secuencia de uno o más treses. El lenguaje L_2 se define entonces como un subconjunto del conjunto anterior: sólo pertenecerán a L_2 las cadenas de C en las que haya más dígitos pares que impares. Así, las cadenas 12223 y 11222222233 pertenecerán a L_2, pero no 1223 ó 111233 (ni 1222 ó 1234444, que ni siquiera pertenecen a C).

Asume que tu especificación léxica define tres categorías, **uno**, **dos** y **tres**, que se emiten y no tienen atributos; asume, además, la siguiente gramática:

$$\begin{aligned}
\langle \mathsf{S} \rangle &\rightarrow \langle \mathsf{I} \rangle\ \langle \mathsf{P} \rangle\ \langle \mathsf{I} \rangle \\
\langle \mathsf{I} \rangle &\rightarrow \textbf{uno}\ \langle \mathsf{I} \rangle \mid \textbf{tres}\ \langle \mathsf{I} \rangle \mid \lambda \\
\langle \mathsf{P} \rangle &\rightarrow \textbf{dos}\ \langle \mathsf{M} \rangle \\
\langle \mathsf{M} \rangle &\rightarrow \textbf{dos}\ \langle \mathsf{M} \rangle \mid \lambda
\end{aligned}$$

A continuación, debes llevar a cabo los siguientes pasos:

A. Añade acciones semánticas a la gramática anterior de modo que el esquema de traducción resultante, mientras se lleva a cabo un análisis LL(1) de la entrada, pueda hacer los cálculos necesarios para acabar obteniendo en un atributo $\langle \mathsf{S} \rangle.ok$ un valor lógico que indique si la cadena de entrada pertenece o no a L_2. El esquema de traducción debe cumplir los siguientes requisitos:

- En las acciones semánticas no ha de utilizarse ningún objeto global.
- Variables locales y atributos sólo pueden ser de tipo entero o lógico.

B. Construye un árbol de análisis para la siguiente secuencia de categorías léxicas:

uno tres uno dos dos dos

C. Vuelve a dibujar el árbol anterior, pero esta vez añade a cada nodo del árbol los atributos que calcularía tu esquema de traducción.

Ejercicio 9 — Valor: 5,00 puntos

Con el objetivo de que diseñes un analizador para un determinado lenguaje de expresiones, L_3, se te proporcionan las siguientes especificaciones. En primer lugar, una especificación léxica:

Categoría léxica	Expresión regular	Atributos	Acciones
literal	`[0-9]+`	*valor*	Calcular valor y emitir
apar	`\(`	—	Emitir
cpar	`\)`	—	Emitir
opmax	`&`	—	Emitir
oppos	`[!?]`	*lexema*	Copiar lexema y emitir
opmin	`x`	—	Emitir

Como modelo del nivel sintáctico del lenguaje, se te proporciona la siguiente gramática, G_0:

$$\begin{aligned}
\langle \mathsf{E} \rangle &\rightarrow \textbf{literal} \\
\langle \mathsf{E} \rangle &\rightarrow \langle \mathsf{E} \rangle\ \textbf{oppos} \\
\langle \mathsf{E} \rangle &\rightarrow \langle \mathsf{E} \rangle\ \textbf{opmax}\ \langle \mathsf{E} \rangle \\
\langle \mathsf{E} \rangle &\rightarrow \langle \mathsf{E} \rangle\ \textbf{opmin}\ \langle \mathsf{E} \rangle \\
\langle \mathsf{E} \rangle &\rightarrow \textbf{apar}\ \langle \mathsf{E} \rangle\ \textbf{cpar}
\end{aligned}$$

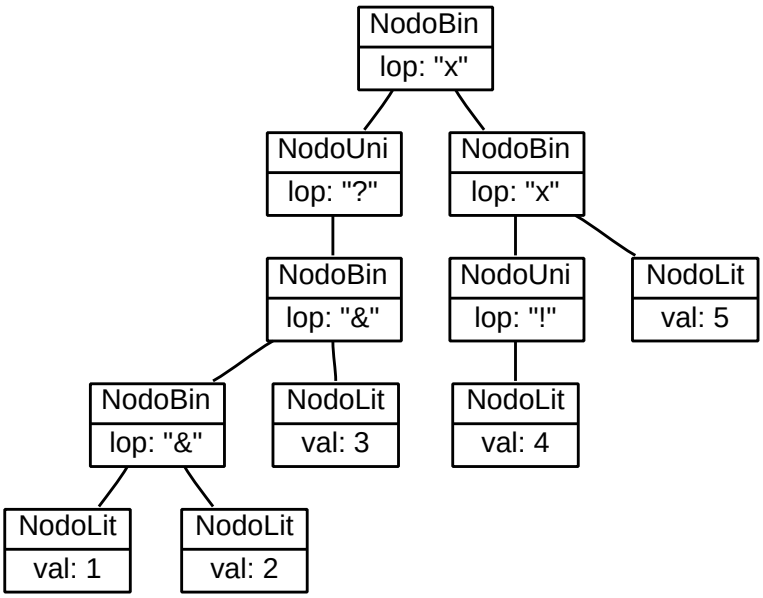
Además, los árboles AST deben construirse utilizando los nodos siguientes:

Nombre	Significado	Atributos	Hijos
NodoLit	Un literal	*val*: valor	—
NodoUni	Una operación unaria	*lop*: lexema del operador	*exp*: único operando
NodoBin	Una operación binaria	*lop*: lexema del operador	*izq* y *der*: operandos

Finalmente, para poder construir los árboles AST correctamente, es necesario que conozcas los diferentes niveles de precedencia en los que se agrupan los operadores de L_3, así como la asociatividad de los que son binarios:

- *Máxima prioridad.* La del operador `&`, asociativo por la izquierda.
- *Prioridad media.* La de los operadores posfijos `!` y `?`.
- *Mínima prioridad.* La del operador `x`, asociativo por la derecha.

Observa, por ejemplo, que la expresión `1&2&3?x4!x5` sería una cadena válida de L_3 para la cual el analizador del lenguaje debería construir el siguiente AST:



Lo que se te pide es que escribas un esquema de traducción para L_3 siguiendo los siguientes pasos:

Paso 1: Sustituye la gramática G_0 por otra, G_1, que genere el mismo lenguaje, sea RLL(1) y tenga como símbolo inicial el no terminal ⟨E⟩.

Paso 2: Añade acciones semánticas a G_1 de modo que el esquema de traducción resultante sintetice el AST correspondiente a la cadena de entrada en el atributo ⟨E⟩.*ast* de la raíz del árbol de análisis. No puedes utilizar ningún objeto global y, para crear los diferentes nodos del AST, dispones de las funciones constructoras cuyo perfil se especifica a continuación:

- `HazNodoLit(val)`.
- `HazNodoUni(lop,exp)`.
- `HazNodoBin(lop,izq,der)`.

Ejercicio 10 — Valor: 0,75 puntos

Independientemente de qué se pretenda calcular en el atributo *z* del símbolo inicial de la gramática, el esquema de traducción siguiente presenta tres errores evidentes:

$$
\begin{array}{lll}
\langle \mathsf{S} \rangle_1 \rightarrow \langle \mathsf{A} \rangle\ \langle \mathsf{B} \rangle\ \langle \mathsf{S} \rangle_2 & \{\ \langle \mathsf{S} \rangle_1.z := \langle \mathsf{A} \rangle.n + \langle \mathsf{S} \rangle_2.z\ \} \\
\langle \mathsf{S} \rangle \rightarrow \lambda & \\
\langle \mathsf{A} \rangle \rightarrow \textbf{pa}\ \langle \mathsf{A} \rangle\ \langle \mathsf{C} \rangle & \{\ \langle \mathsf{A} \rangle.n := \langle \mathsf{A} \rangle.n + 3 \times \langle \mathsf{C} \rangle.n\ \} \\
\langle \mathsf{A} \rangle \rightarrow \textbf{pe} & \{\ \langle \mathsf{A} \rangle.n := 5;\ \langle \mathsf{S} \rangle.z := 0\ \} \\
\langle \mathsf{B} \rangle \rightarrow \textbf{pi} & \\
\langle \mathsf{B} \rangle \rightarrow \textbf{po} & \\
\langle \mathsf{C} \rangle \rightarrow \textbf{pu} & \{\ \langle \mathsf{C} \rangle.n := 7\ \}
\end{array}
$$

Señala esos errores en el esquema e indica concisa y claramente en qué consisten.

Ejercicio 11 — Valor: 2,00 puntos

Para poder gestionar los tipos de las expresiones modeladas mediante la gramática

$$\begin{aligned}
\langle \mathsf{E} \rangle &\rightarrow \langle \mathsf{X} \rangle\ \langle \mathsf{M} \rangle \\
\langle \mathsf{X} \rangle &\rightarrow \textbf{litV} \\
\langle \mathsf{X} \rangle &\rightarrow \textbf{litW} \\
\langle \mathsf{X} \rangle &\rightarrow \textbf{apar}\ \langle \mathsf{E} \rangle\ \textbf{cpar} \\
\langle \mathsf{M} \rangle &\rightarrow \langle \mathsf{O} \rangle\ \langle \mathsf{E} \rangle \\
\langle \mathsf{M} \rangle &\rightarrow \lambda \\
\langle \mathsf{O} \rangle &\rightarrow \textbf{opA} \\
\langle \mathsf{O} \rangle &\rightarrow \textbf{opB}
\end{aligned}$$

habrá que tener en cuenta lo siguiente:

- Los operadores binarios **opA** y **opB** son asociativos por la derecha.
- Hay dos tipos, *V* y *W*, que cuentan con sus correspondientes literales, **litV** y **litW**.
- Habrá que manejar también un tipo especial, *ERROR*, que será el asociado a las expresiones con algún error de tipos.
- El operador **opA** exige que sus dos operandos sean del mismo tipo y, entonces, ése será el tipo del resultado; en otro caso, se producirá un error de tipos.
- El operador **opB** produce un resultado de tipo *W* independientemente de si sus operandos son de tipo *V* o *W*; sólo presentará error de tipos si alguno de sus operandos lo presenta.
- Encerrar una expresión entre paréntesis (componentes **apar** y **cpar**) no afecta a su tipo.

Observa que, por tanto, cada una de las tres expresiones que a continuación se muestran como ejemplo tiene un tipo distinto:

- La expresión **litV opA litV** es de tipo *V*.
- La expresión **litW opA litV opB litV** es de tipo *W*.
- La expresión **apar litW opA litV cpar opB litV** es de tipo *ERROR*.

Lo que tú debes hacer es añadir acciones semánticas a la gramática anterior para que el esquema de traducción resultante pueda sintetizar, mientras lleva a cabo el análisis LL(1) de una expresión, un atributo $\langle \mathsf{E} \rangle.t$ que indique el tipo de esa expresión (*V*, *W* o *ERROR*) en el nodo raíz del correspondiente árbol de análisis.

Además, has de observar las siguientes restricciones a la hora de construir el esquema de traducción:

- No puedes utilizar ningún objeto global.
- Sólo se permiten acciones semánticas en el extremo derecho de la parte derecha de cada producción, lo que obliga a que todos los atributos utilizados deban ser sintetizados.

Nota final

En esta recopilación, los ejercicios han sido reproducidos en el mismo orden en el que fueron apareciendo en los sucesivos exámenes de la asignatura. Para resolverlos aproximadamente en orden de dificultad creciente, se propone seguir esta secuencia: **10**, **7**, **4**, **1**, **11**, **6**, **2**, **5**, **8**, **3** y **9**.