Algunos ejercicios sobre modelado sintáctico en exámenes de Compiladores e intérpretes

IG29: Compiladores e intérpretes

Séptima sesión de teoría

Ejercicio 1 — Valor: 2,00 puntos

Considera el lenguaje de las listas que cumplen las siguientes condiciones:

- La lista empieza y termina con un componente léxico delimitador **delim**.
- No existe ningún separador entre los elementos de la lista.
- El número de elementos de la lista ha de ser impar.
- Cada elemento de la lista es, a su vez, otra lista, o bien un componente léxico **elsimple**.

Y, a continuación, haz lo siguiente:

A. Modela, mediante una gramática incontextual con símbolo inicial ⟨Ll⟩, el lenguaje anteriormente descrito.

B. Modela ese mismo lenguaje mediante una única producción con construcciones regulares en su parte derecha.

Ejercicio 2 — Valor: 1,50 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`...

Supón, además, 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.

Ahora debes modelar L_1 con herramientas propias del nivel sintáctico:

A. Modela L_1 mediante una gramática G_{ER}, utilizando una única producción con parte izquierda ⟨S⟩ y construcciones regulares en su parte derecha.

B. Obtén una gramática G_{GI}, incontextual y sin partes derechas regulares, equivalente a la anterior.

Ejercicio 3 — Valor: 2,00 puntos

Diremos que un *fichero tabulado* se compone de líneas, acabadas cada una en un componente **eol** y con un cierto número no nulo de posiciones, donde cada posición consiste en un tabulador (componente **tab**) posiblemente seguido de un elemento (componente **ele**) que cabe omitir.

Teniendo en cuenta la anterior definición de fichero tabulado, modela cada uno de los dos lenguajes que se describen a continuación mediante un gramática incontextual y no ambigua:

A. El lenguaje de los ficheros tabulados de exactamente dos líneas. Un ejemplo de cadena de este lenguaje podría ser el siguiente: **tab tab tab ele tab ele eol tab ele eol**.

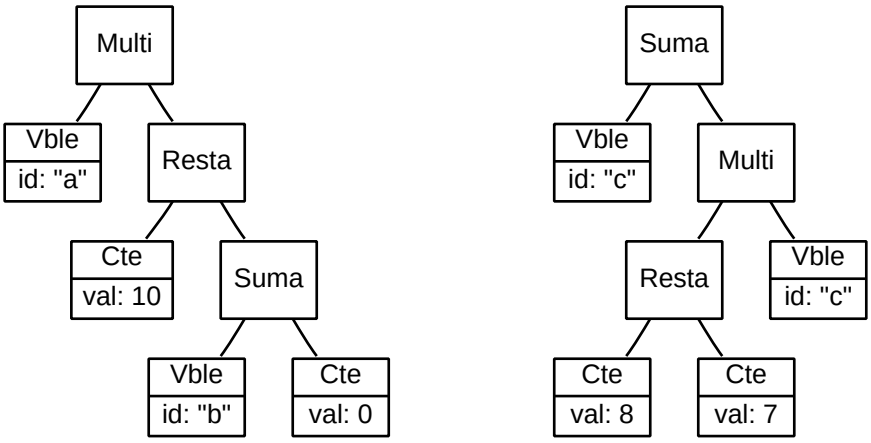
B. Un subconjunto del lenguaje anterior: el que obliga a que las dos líneas del fichero contengan el mismo número de posiciones (y, por tanto, de tabuladores). Por ejemplo, el fichero tabulado **tab ele tab ele tab ele eol tab tab ele tab eol** tiene exactamente tres posiciones en cada una de sus dos líneas.

Ejercicio 4 — Valor: 3,00 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:



Se trata de que sigas los pasos que se detallan a continuación para modelar léxica y sintácticamente el lenguaje de expresiones *LEDi3*:

A. Dota al lenguaje de una especificación léxica adecuada construyendo la correspondiente tabla con columnas **Categoría léxica**, **Expresión regular**, **Emitir u omitir** y **Atributos**. No incluyas en la especificación atributos que no vayan a ser necesarios para la posterior construcción de los correspondientes árboles de sintaxis abstracta.

B. Utiliza las categorías léxicas de tu especificación anterior para modelar, mediante una gramática incontextual con símbolo inicial ⟨Cad⟩ (y no lo olvides: sin partes derechas regulares), el nivel sintáctico de *LEDi3*. La gramática no debe ser ambigua, pero no se exige que sea LL(1), sino que refleje correctamente la precedencia y asociatividad de los operadores del lenguaje.

Ejercicio 5 — Valor: 2,00 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**.

Teniendo en cuenta todo lo anterior, haz lo que se te pide a continuación:

A. Modela, mediante una gramática incontextual con símbolo inicial ⟨Matriz⟩ (y no lo olvides: sin partes derechas regulares), el nivel sintáctico de ese lenguaje de matrices.

B. Modela ese mismo nivel sintáctico pero, esta vez, utilizando una gramática con una única producción. En la parte derecha de esta producción sí puedes utilizar construcciones regulares, pero sólo |, * y ? (y no la clausura positiva +).

Ejercicio 6 — Valor: 1,50 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).

Se trata de que modeles L_2 asumiendo que tu especificación léxica define tres categorías, **uno**, **dos** y **tres**, que se emiten y no tienen atributos. Para modelar el lenguaje puedes elegir una cualquiera de las dos opciones siguientes:

- Diseñar una gramática incontextual que genere el lenguaje L_2.
- Diseñar un esquema de traducción dirigido por la sintaxis que cumpla los siguientes requisitos:
  - La gramática ha de ser RLL(1) y generar, al menos, todas las cadenas de L_2.
  - En las acciones semánticas no ha de utilizarse ningún objeto global.
  - Variables locales y atributos sólo pueden ser de tipo entero.
  - El análisis de cualquier cadena generada por la gramática y que no pertenezca a L_2 debe dar lugar a una llamada a la función predefinida `rechaza_cadena`.

Ejercicio 7 — Valor: 4,00 puntos

Definamos la estructura de un nuevo lenguaje, *ListO*. Para ello, comenzamos diciendo que un programa *ListO* consistirá en una secuencia de elementos, secuencia a la que llamaremos lista principal del programa. Los elementos de esta lista principal podrán ser definiciones o sentencias:

- Si hay definiciones, éstas deberán aparecer antes que ninguna sentencia de la lista principal.
- Será obligatorio que haya al menos una sentencia en la lista principal.

Una definición *ListO* irá encerrada dentro de un par de paréntesis (componentes **apar** y **cpar**) y constará de tres elementos separados por comas (componentes **coma**), por este orden: la palabra clave **Define**, un identificador (componente **ident**) y una sentencia.
Una sentencia *ListO* también irá encerrada entre paréntesis y constará de un número variable de elementos (al menos uno) separados por comas; la naturaleza del primer elemento determinará qué elementos adicionales se necesitan:

- Palabra clave **Lee**: Debe haber un único elemento adicional, que será un identificador.
- Palabra clave **Asigna**: Dos elementos adicionales, un identificador seguido de una expresión.
- Palabra clave **Muestra**: Uno o más elementos adicionales, que serán expresiones.
- Un identificador: Ningún elemento adicional.

Finalmente, una expresión *ListO* presentará una de las siguientes estructuras:

- Podrá tratarse, simplemente, de un literal (componente **lit**) o de un identificador.
- Podrá consistir en una lista entre paréntesis de tres elementos separados por comas: un operador (componente **idop**) seguido de dos expresiones *ListO*.

Así, por ejemplo, podemos considerar que el programa

```
(Define, incrementa_a, (Asigna,  a, (+, a, 1)))
(Define, muestra_a_b,  (Muestra, a,
                                 b,
                                 (*, a, b),
                                 (+, (*, a, a),
                                     (*, b, b))))
(Lee, a) (Lee, b) (muestra_a_b)
(incrementa_a)    (muestra_a_b)
(incrementa_a)    (muestra_a_b)
```

sigue la sintaxis descrita si asumimos la especificación léxica siguiente:

Categoría léxica	Expresión regular	Atributos	Acciones
blancos	`[ \t\n]+`	—	Omitir
apar	`\(`	—	Emitir
cpar	`\)`	—	Emitir
coma	`,`	—	Emitir
Define	`Define`	—	Emitir
Lee	`Lee`	—	Emitir
Asigna	`Asigna`	—	Emitir
Muestra	`Muestra`	—	Emitir
ident	`[a-z_]+`	*lexema*	Copiar lexema y emitir
lit	`[0-9]+(\.[0-9]+)?`	*valor*	Calcular valor y emitir
idop	`[-+*/]`	*lexema*	Copiar lexema y emitir

Debes modelar el nivel sintáctico de *ListO* de dos formas diferentes:

A. Primero, mediante una gramática con partes derechas regulares en las que, si lo deseas, puedes utilizar clausuras positivas (además de los operadores habituales de disyunción, opcionalidad y clausura).

B. Después, debes escribir una gramática incontextual LL(1) que modele el mismo lenguaje.

Ejercicio 8 — Valor: 1,00 puntos

Modela mediante una gramática no ambigua el lenguaje de todas las secuencias de puntos geométricos que cumplen las condiciones siguientes:

- Las secuencias válidas en el lenguaje tienen al menos dos puntos.
- El lenguaje no utiliza ningún componente léxico para separar entre sí los puntos de una secuencia.
- Cada punto viene delimitado por un par de paréntesis: el abierto **apar** y el cerrado **cpar**.
- Entre los paréntesis, el punto se especifica como una secuencia de literales reales **real** separados entre sí por componentes **coma**.
- Cada punto puede pertenecer a $\mathbb{R}^2$ o a $\mathbb{R}^3$, esto es, estará especificado mediante dos o tres literales.

Así, un ejemplo de secuencia del lenguaje sería la siguiente:

$$\underbrace{\textbf{apar real coma real coma real cpar}}_{\text{un punto de } \mathbb{R}^3}\ \underbrace{\textbf{apar real coma real cpar}}_{\text{uno de } \mathbb{R}^2}\ \underbrace{\textbf{apar real coma real cpar}}_{\text{y otro de } \mathbb{R}^2}$$

Utiliza una gramatica incontextual o, si lo prefieres, una con partes derechas regulares. En cualquier caso, no está permitido que hagas uso de clausuras positivas.

Ejercicio 9 — Valor: 1,25 puntos

Modela mediante una gramática no ambigua el lenguaje de todas las listas que cumplan las condiciones siguientes:

- Una lista de este lenguaje estará formada por tres o más elementos.
- Además, estará delimitada por un componente inicial, **ini**, y otro final, **fin**.
- Los elementos de cada lista estarán separados entre sí por componentes **sep**.
- Cada elemento consistirá en un componente **id** que, opcionalmente, podrá ir seguido de una lista.

Así, un ejemplo de lista del lenguaje, con cuatro elementos, podría ser el siguiente:

$$\textbf{ini id sep id } \overbrace{\textbf{ini id } \underbrace{\textbf{ini id sep id sep id sep id fin}}_{\text{con sublista en su primer elemento}}\ \textbf{sep id sep id sep id fin}}^{\text{sublista de cuatro elementos en el segundo elemento de la lista principal...}}\ \textbf{sep id sep id } \overbrace{\textbf{ini id sep id sep id fin}}^{\text{sublista de tres elementos}}\ \textbf{fin}$$

Utiliza para el modelado una gramatica incontextual o, si lo prefieres, una con partes derechas regulares. En cualquier caso, no está permitido que hagas uso de clausuras positivas.

Ejercicio 10 — Valor: 1,50 puntos

Supón que la sintaxis del lenguaje de programación *SePT06* viene dada por la gramática G_0:

⟨Prog⟩ → ⟨Sent⟩
⟨Sent⟩ → **id asigna** ⟨Expr⟩ **pyc**
⟨Sent⟩ → **IF** ⟨Expr⟩ **THEN** ⟨Sent⟩
⟨Sent⟩ → **FOREVER DO** ⟨Sent⟩
⟨Sent⟩ → **BREAK pyc**
⟨Sent⟩ → **BEGIN** ⟨Lista⟩ **END**
⟨Lista⟩ → ⟨Sent⟩ ⟨Lista⟩ | λ
⟨Expr⟩ → ⟨Base⟩ ⟨Más⟩
⟨Más⟩ → **op** ⟨Base⟩ ⟨Más⟩ | λ
⟨Base⟩ → **id** | **lit** | **apar** ⟨Expr⟩ **cpar**

donde cabe resaltar que las sentencias del lenguaje son de cinco tipos distintos: la asignación de una expresión a una variable, una estructura de control condicional `IF`, un bucle infinito `FOREVER`, una sentencia `BREAK` para salir incondicionalmente del bucle y la sentencia compuesta `BEGIN...END`.

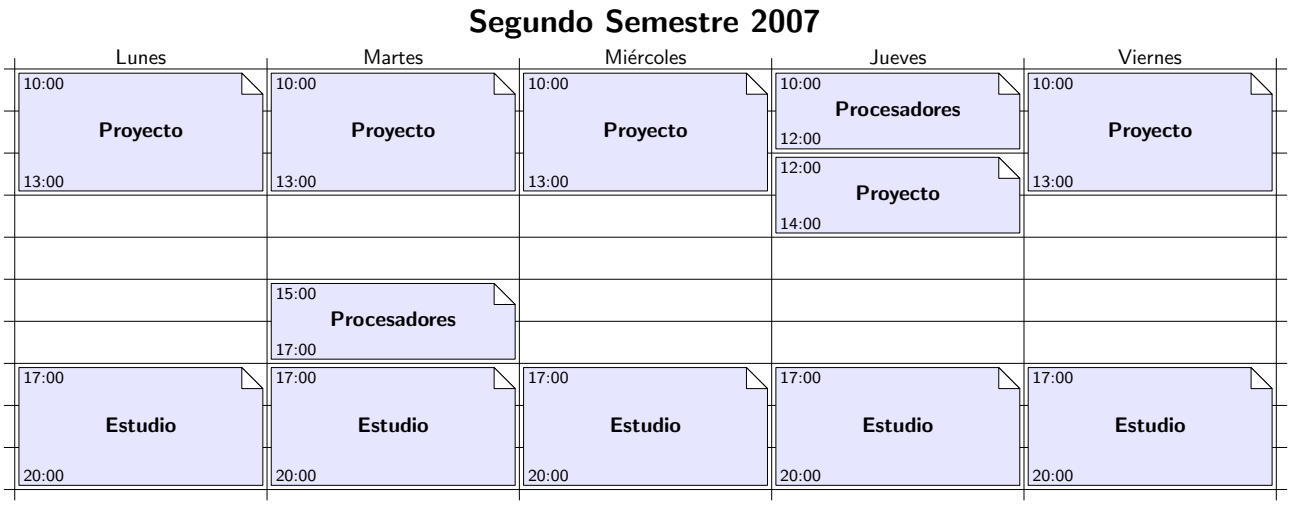
Observa que, aunque la gramática G_0 no lo impide, no tiene sentido utilizar una sentencia `BREAK` fuera de un bucle; se trata, pues, de que impongas la correspondiente restricción adicional (es decir, una sentencia `BREAK` sólo podrá ser considerada correcta si aparece dentro de un bucle `FOREVER`) mediante una nueva gramática no ambigua que genere el lenguaje de los programas *SePT06* sin sentencias `BREAK` incorrectas.

Ejercicio 11 — Valor: 2,00 puntos

Supón que se está diseñando una aplicación para la gestión de horarios personales y que tú eres el encargado de formalizar la sintaxis de los ficheros de horarios que aceptará como entrada tal aplicación. Cada fichero se supone que especifica el horario semanal de una persona, incluye un título y ubica una serie de actividades en diferentes franjas horarias de la semana. Por ejemplo,

```
Título: "Segundo Semestre 2007"
Días: L-V
Horas: 10-20
Actividades: "Procesadores" M * 15-17, J * 10-12;
             "Proyecto" L-X + V * 10-13, J * 12-14;
             "Estudio" * 17-20;
```

podría ser el contenido de un fichero que especificara este horario:



En general, de la sintaxis de los ficheros de entrada puede decirse lo siguiente:

- Cada fichero debe especificar un horario semanal y se divide en cuatro bloques: título, días, horas y actividades.
- El bloque *título* es obligatorio y consta de la palabra clave **Título**, un componente **dp** (dos puntos) y un literal **cadena** cuyo valor especifica el título del horario.
- El bloque *días* es opcional y, de aparecer, constará de la palabra clave **Días**, un componente **dp** y un rango de días. Este rango constará a su vez de dos literales **día** separados por un componente **guion** y servirá para restringir qué días de la semana deben incluirse en el horario, en vez de los siete.
- El bloque *horas* también es opcional y, de aparecer, constará de la palabra clave **Horas**, **dp** y un rango de horas, especificado éste mediante dos literales **hora** separados por un componente **guion**; análogamente al caso anterior, este rango servirá para restringir qué horas, en vez de las veinticuatro de cada día, deben incluirse en el horario.
- El bloque más complejo, *actividades*, es obligatorio y consta de la palabra clave **Actividades**, **dp** y una secuencia posiblemente vacía de actividades, donde de cada actividad se especificará su nombre, mediante un literal **cadena**, y en qué franjas horarias se llevará a cabo. Cada especificación de actividad concluirá con un terminador **pyc** (punto y coma).
- La especificación de franjas horarias de una actividad consta de uno o más elementos, separados por componentes **coma**; cada uno de estos elementos tendrá, a su vez la siguiente estructura: un conjunto de días, un componente **por** y un rango de horas.
- Un conjunto de días puede omitirse para indicar que se está haciendo referencia a todos los días incluidos en el horario; de no ser así, el conjunto se especificará mediante una secuencia de uno o más elementos, separados por componentes **más**, y cada uno de estos elementos será un literal **día** o bien un rango de días.

Teniendo en cuenta todo lo anterior, modela formalmente, mediante una gramática no ambigua, la sintaxis de los ficheros de entrada de la aplicación de horarios.