Teoría de Lenguajes

Práctica 7
(Escritura de gramáticas libres de contexto)

1. Una lista en el lenguaje PROLOG se puede representar como una secuencia de elementos encerrados entre corchetes y separados por comas. Los elementos de la lista pueden ser a su vez listas. De esta manera, los siguientes son ejemplos de listas:

   A_1 = `[a]` A_2 = `[]`
   A_3 = `[a, [b, c], d]` A_4 = `[[], [a, [a], b, [[]]]]`

   Llamaremos L al lenguaje sobre el alfabeto $\Sigma = \{$ `[` ; `]` ; `,` ; $id\}$ formado por las listas recién descriptas. Se pide dar una gramática no ambigua para L.

2. Consideramos una sintaxis simplificada para expresionenes del lenguaje Common LISP: una expresión puede ser un átomo o una lista. Los átomos pueden ser símbolos, números, o cadenas (terminales `sym`, `num` y `str`, respectivamente). Una lista es una secuencia de expresiones encerradas entre paréntesis: `(` y `)`. Las listas pueden ser vacías. Ejemplos de expresiones válidas pueden ser entonces:

   ```
   num
   ()
   ( sym () num str )
   ( ( num ) sym ( sym num str () ) )
   ```

   Llamaremos L al lenguaje sobre el alfabeto $\Sigma = \{$`(`, `)`, `sym`, `num`, `str`$\}$ formado por las listas recién descriptas. Se pide dar una gramática no ambigua para L.

3. Una fórmula química es una manera concisa de expresar información sobre los átomos que constituyen un compuesto. Cada elemento es identificado por su símbolo químico y la cantidad de átomos de cada elemento es indicada por un subíndice, si es mayor que uno. Por ejemplo, el metano, una molécula simple compuesta por un atomo de carbono unido a cuatro de hidrógeno, tiene la fórmula química CH_4. Si un ion se repite más de una vez, esto se puede expresar encerrándolo entre paréntesis y agregando un subíndice indicando la cantidad de veces que se repite. Por ejemplo, el sulfato férrico está compuesto por dos átomos de hierro y tres iones sulfato, cada uno de los cuales se compone de un átomo de azufre y cuatro

de oxígeno: $Fe_2(SO_4)_3$. De esta manera, los siguientes son ejemplos de fórmulas químicas:

Oxígeno: O_2 Agua: H_2O
Ferrocianuro Férrico: $Fe_2(Fe(CN)_3)_3$

Llamaremos L al lenguaje sobre el alfabeto $\Sigma = \{$`(`, `)`, *elem*, *num*$\}$ formado por las fórmulas químicas recién descriptas. Se pide dar una gramática no ambigua para L.

4. Dar una gramática no ambigua para expresiones aritméticas sobre identificadores con suma, resta, producto, división y paréntesis. Como es usual, todas las operaciones son asociativas a izquierda. La prioridad de la suma y la resta es menor que la del producto y la división. El símbolo del producto se puede omitir.
   Dar el árbol de derivación de la expresión `id - id id / id * id + id`.

5. La siguiente gramática representa un fragmento de las expresiones válidas en el lenguaje de programacion C:

   $G = <\{$`id`, `?`, `:`, `+`$\}, \{E\}, P, E>$, con P:

   $$E \longrightarrow E \texttt{ ? } E \texttt{ : } E$$
   $$E \longrightarrow E \texttt{ + } E$$
   $$E \longrightarrow \texttt{id}$$
   $$E \longrightarrow \texttt{(} E \texttt{)}$$

   (a) Para las cadenas $\alpha_1 =$ `id ? id : id + id ? id : id` y $\alpha_2 =$ `id ? id ? id + id : id : id`, dar todos sus árboles de derivación.

   (b) Teniendo en cuenta que el operador ternario condicional `? :` es asociativo a derecha, y tiene menor precedencia que el operador `+`, que es asociativo a izquierda, dar una gramática no ambigua para $L(G)$. Dar los árboles de derivación resultantes para α_1 y α_2.

6. Consideraremos una forma simplificada para la sintaxis de expresiones del lenguaje Smalltalk. Las subexpresiones "basicas" son los identificadores (terminal *id*) y literales de enteros (terminal *int*). Hay tres tipos de subexpresiones compuestas:

   - Expresiones unarias: un operador unario es un identificador (*id*) y se escribe *después* de la subexpresión a la cual se aplica. Por ejemplo, `x sqrt` representa la raíz cuadrada de x.
   - Expresiones binarias: hay un gran número de operadores binarios, que representaremos mediante el símbolo terminal *binOp*. Se escriben en forma infija, y se evalúan estrictamente de izquierda a derecha (es decir que no hay precedencia entre ellos). Entonces, `x+y*z` significa lo mismo que `(x+y)*z`
   - Expresiones con palabras clave: tienen la forma $e_0 k_1 e_1 \ldots k_n e_n$, es decir, una subexpresión e_0 y luego palabras clave k_i alternando con subexpresiones e_i. Cada palabra clave k_i se asocia con su expresión

correspondiente e_i. Las palabras clave son reconocidas por el analizador léxico y se representan por el terminal *key*. En la cadena de entrada, se ven como un identificador seguido por dos puntos. Un ejemplo de expresión con palabras clave es: `a at: 3 put: (x+y)`

Precedencia: a menos que se usen paréntesis, primero se aplican siempre los operadores unarios, después los binarios, y después se evalúan las expresiones con palabras clave. Entonces, la expresión α:

```
a at: x+5 put: (b+c / d sqrt) truncate
```

tiene el mismo significado que la expresión α':

```
a at: (x+5) put: (((b+c) / (d sqrt)) truncate)
```

Llamaremos L al lenguaje sobre el alfabeto $\Sigma = \{id, int, binOp, key, (,)\}$ formado por las expresiones recién descriptas.

(a) Dar una gramática no ambigua G para L que respete las reglas de precedencia y asociatividad descriptas.

(b) Dar el árbol de derivación producido por la gramática para la expresión α.