E. T. S. DE INGENIERÍA INFORMÁTICA
Departamento de Estadística, I.O. y Computación
Teoría de Autómatas y Lenguajes Formales

PRACTICA 6: Eliminación de símbolos y producciones inútiles en una Gramática Independiente del Contexto

6.1. Requisito de codificación

Comentarios:
La utilización de comentarios es fundamental a la hora de clarificar en la mayor medida posible un código fuente.

Recomendamos colocar siempre comentarios en las declaraciones, tanto de variables como de atributos de las clases.

Recordemos que el mejor comentario es en sí mismo el propio identificador que elijamos para la entidad en cuestión (variable, atributo, método, etc.) pero en todo caso siempre se puede aclarar su significado poniendo un breve comentario a la derecha de la declaración.

La regla a seguir a la hora de poner comentarios es *Indique el motivo por el que se hace algo, no lo que se hace*. Recuerde que cualquiera que lea su código fuente debería conocer el lenguaje de programación. Por tanto, los comentarios no deberían ser una traducción al castellano de lo que indica el código fuente:

```
x = x + 1; // Se incrementa x en una unidad (INCORRECTO)
x = x + 1; // Para considerar el primer caso (CORRECTO)
```

Recuerde también que un código *hipercomentado* es tan malo como uno que carece por completo de comentarios. Podríamos estimar que un comentario cada 20 líneas de código podría ser un buen promedio, aunque ello depende mucho del código del que se trate.

6.2. Introducción

Dada una gramática independiente del contexto (de ahora en adelante, una gramática), G $\equiv$ (V, Σ, S, P), un símbolo $X \in (V \cup \Sigma)$ se dice que es útil si y sólo si existe una derivación de la forma:
$S \Rightarrow^* \alpha X \beta \Rightarrow^* \omega \in \Sigma^*$
es decir, se han de dar dos condiciones:

1. que a partir del símbolo X se pueda derivar una cadena de símbolos exclusivamente terminales.
2. que el símbolo X aparezca en una forma sentencial.

Una regla de producción de una gramática se dice útil si lo son todos sus símbolos.

Las figuras 6.1 y 6.2 describen en pseudocódigo un algoritmo para eliminar símbolos y producciones inútiles en una gramática. El algoritmo se divide en dos etapas: la primera de ellas elimina los símbolos que no cumplen la primera condición y la segunda elimina los símbolos que no cumplen la segunda. Estas etapas han de aplicarse en el orden expuesto, para obtener el resultado deseado. En la figura 6.2 V" y J son conjuntos de

$V' = \emptyset$
for cada producción de la forma $A \rightarrow \omega$ **do**
  $V' = V' \cup \{A\}$
while (V' cambie) **do**
  for cada producción de la forma $B \rightarrow \alpha$ **do**
    if (todas las variables de α pertenecen a V')
      $V' = V' \cup \{B\}$
Eliminar todas las variables ($C \in V$) que estén en V y no en V'
Eliminar todas las producciones donde aparezca una variable de las eliminadas en el paso anterior

Figura 6.1: Primera etapa del algoritmo de eliminación de símbolos y producciones inútiles

símbolos no terminales (de V), T' es un conjunto de símbolos terminales (de Σ) y J es el conjunto de las variables que están pendientes de analizar.

6.3. Práctica

La práctica consiste en desarrollar un programa en C++ que dada una gramática independiente del contexto, le aplique el algoritmo de eliminación de producciones y símbolos inútiles.

Los ficheros que se utilizarán para representar las gramáticas tanto de entrada como de salida, tendrán el mismo formato que los que ya se han utilizado en una práctica anterior.

$J = \{S\}$
$V'' = \{S\}$
$T' = \emptyset$
while ($J \neq \emptyset$) **do**
  Extraer un no terminal A de J: $J = J - \{A\}$
  for cada producción de la forma $A \rightarrow \alpha$
    begin
    for cada no-terminal B en α
      if ($B \notin V''$) **then**
        begin
        $J = J \cup \{B\}$
        $V'' = V'' \cup \{B\}$
        end
    Poner todos los terminales de α en T'
    end
Eliminar todas las variables que no estén en V" y todos los terminales que no estén en T'.
Eliminar todas las producciones donde aparezca un símbolo o variable de los eliminados.

Figura 6.2: Segunda etapa del algoritmo de eliminación de símbolos y producciones inútiles