Universidad de La Laguna

ESIT – INFORMÁTICA
Departamento de Ingeniería Informática y de Sistemas

Computabilidad y Algoritmia
Curso 2019-2020

PRACTICA 9: Forma Normal de Chomsky de una Gramática Independiente del Contexto

9.1. Objetivos

- Consolidar los conocimientos adquiridos sobre Gramáticas.
- Estudiar y practicar el algoritmo de transformación de una gramática a su forma normal de Chomsky.
- Ampliar las funcionalidades definidas en la clase que se ha desarrollado para representar Gramáticas.
- Profundizar en las capacidades de diseñar y desarrollar programas orientados a objetos en C++.

9.2. Introducción

En teoría de lenguajes formales se dice que una gramática independiente del contexto $G \equiv (V, \Sigma, S, P)$ está en forma normal de Chomsky (en honor a quien propuso este formato, CNF, *Chomsky Normal Form*) si:

- V sólo contiene símbolos útiles.
- Todas las producciones de G tienen una de las formas:
  $A \to BC$
  $A \to a$

Si $\epsilon \in L(G)$ se permite además una única producción $S \to \epsilon$ y en este caso no se permite que el símbolo de arranque figure en la parte derecha de ninguna regla de producción.

Al margen del interés teórico que tiene la forma normal de Chomsky (se utiliza por ejemplo en la demostración del lema del bombeo para lenguajes independientes del contexto), la conversión a CNF se utiliza como paso previo de otros algoritmos, como el algoritmo de análisis sintáctico CYK [2].

Cualquier gramática independiente del contexto puede ser transformada en una equivalente escrita en forma normal de Chomsky. Por otra parte, se puede demostrar que en una Gramática escrita en CNF, la derivación de una cadena de *n* símbolos tiene exactamente *2n -1* pasos de derivación. Esta propiedad permite determinar si una cadena $w \in L(G)$ mediante una inspección exhaustiva de todas las derivaciones.

El Algoritmo 1 aplicado a una CFG permite su transformación a FNC. El algoritmo toma como entrada una gramática que no contenga símbolos ni producciones inútiles, unitarias ni vacías, y produce como salida una gramática equivalente a la de entrada, pero escrita en Forma Normal de Chomsky.

```
1  forall (A → X_1 X_2 ... X_n (con n ≥ 2, X_i ∈ (Σ ∪ V)) do
2      forall (X_i) do
3          if (X_i = a ∈ Σ) then
4              Add the production C_a → a ;
5              Replace X_i with C_a in A → X_1 X_2 ... X_n ;
6          end
7      end
8  end
9  forall (A → B_1 B_2 ... B_m (con m ≥ 3, B_i ∈ V) do
10     Add m − 2 non-terminal symbols D_1 D_2 ... D_{m−2};
11     Replace the production A → B_1 B_2 ... B_m with productions:
12     A → B_1 D_1
13     D_1 → B_2 D_2
14     ...
15     D_{m−2} → B_{m−1} B_m
16 end
```

Algorithm 1: Transformación de una CFG a su Forma Normal de Chomsky

Considérese a modo de ejemplo la gramática definida por las siguientes producciones:

$S \to aXbX \mid abX \mid aXb \mid ab$
$X \to aY \mid bY \mid a \mid b$
$Y \to aY \mid bY \mid a \mid b \mid c$

El primer bucle del algoritmo 1 (líneas 1–8) introduce dos nuevos símbolos no terminales C_a y C_b y sus correspondientes producciones, de modo que la gramática se transforma en:

$S \to C_aXC_bX \mid C_aC_bX \mid C_aXC_b \mid C_aC_b$
$X \to C_aY \mid C_bY \mid a \mid b$
$Y \to C_aY \mid C_bY \mid a \mid b \mid c$
$C_a \to a$
$C_b \to b$

Finalmente el segundo bucle del algoritmo (líneas 9–16) transforma la gramática en:

$S \rightarrow C_a D_1 \mid C_a E_1 \mid C_a F_1 \mid C_a C_b$
$X \rightarrow C_a Y \mid C_b Y \mid a \mid b$
$Y \rightarrow C_a Y \mid C_b Y \mid a \mid b \mid c$
$C_a \rightarrow a$
$C_b \rightarrow b$
$D_1 \rightarrow X D_2$
$D_2 \rightarrow C_b X$
$E_1 \rightarrow C_b X$
$F_1 \rightarrow X C_b$

que ya está representada en Forma Normal de Chomsky.

Estas referencias [3] [4] presentan ejemplos adicionales de conversión a CNF.

9.3. Ejercicio práctico

Desarrollar un programa `G2CNF.cpp` que lea un fichero de texto en el que figura la especificación de una Gramática Independiente del Contexto que no contenga símbolos ni producciones inútiles, unitarias ni vacías, y genere otro fichero de texto en el que se especifique una gramática equivalente ($L(G_{in}) = L(G_{out})$) a la de entrada, escrita en Forma Normal de Chomsky.

El comportamiento del programa al ejecutarse en línea de comandos debiera ser:

```
$ ./G2CNF
Modo de empleo: ./G2CNF input.gra output.gra
Pruebe 'G2CNF --help' para más información.
```

Donde `input.gra` y `output.gra` son los ficheros que especifican las gramáticas de entrada y salida respectivamente. La opción `--help` en línea de comandos ha de producir que se imprima en pantalla un breve texto explicativo del funcionamiento del programa.

Los ficheros de especificación de gramáticas son ficheros de texto plano con extensión `.gra` que contienen los elementos definitorios de la Gramática $G \equiv (\Sigma, V, S, P)$ en este orden: símbolos terminales, símbolos no terminales, símbolo de arranque y producciones. El formato de cada uno de estos elementos en el fichero es el siguiente:

1. Símbolos terminales (alfabeto): una línea que contiene N, el número de símbolos en el alfabeto seguida de N líneas, cada una de las cuales contiene un símbolo del alfabeto. Cada símbolo del alfabeto debe ser un único caracter imprimible.

2. Conjunto de símbolos no terminales: una línea que contiene V, el número de símbolos no terminales, seguida de V líneas, cada una de las cuales contiene una cadena alfanumérica sin espacios.

3. Símbolo de arranque: una única línea que contiene el símbolo de arranque, S, de la gramática. Ha de ser uno de los símbolos no terminales relacionados anteriormente.

4. Producciones: una línea que contiene P, el número de producciones de la gramática, seguida por P líneas cada una de las cuales contiene una producción en el formato:

   `A ->` α

   siendo $\alpha \in (\Sigma \cup V)^*$, es decir una secuencia de símbolos terminales y no terminales. La cadena vacía, ϵ se representa mediante el carácter `~` (código ASCII 126).

Todas las líneas de un fichero `.gra` que comiencen con los caracteres `//` corresponden a comentarios, y deben ser ignorados por el programa a la hora de procesar el fichero.

9.4. Rúbrica de evaluación del ejercicio

Se señalan a continuación los aspectos más relevantes (la lista no es exhaustiva) que el profesorado tendrá en cuenta a la hora de evaluar el trabajo que el alumnado presentará en la sesión de evaluación de la práctica:

- Capacidad del programador(a) de introducir cambios en el programa desarrollado.
- El comportamiento del programa debe ajustarse a lo solicitado en el enunciado.
- El programa diseñado ha de seguir el paradigma OOP.
- Se valorará la coherencia e idoneidad del diseño realizado en las clases diseñadas para representar gramáticas.
- Modularidad: el programa ha de escribirse de modo que las diferentes funcionalidades que se precisen sean encapsuladas en métodos cuya extensión textual se mantenga acotada.
- El programa ha de ceñirse al formato de escritura de programas adoptado en las prácticas de la asignatura.
- En la sesión de evaluación de este trabajo, todos los ficheros con códido fuente se han de alojar en un único directorio junto con el fichero Makefile de compilación.
- Todos los atributos de las clases definidas en el proyecto han de tener un comentario descriptivo de la finalidad del atributo en cuestión.
- Se valorará que los comentarios del código fuente sigan el formato de los comentarios de Doxygen.

Si el alumnado tiene dudas respecto a cualquiera de estos aspectos, debiera acudir al foro de discusiones de la asignatura para plantearlas allí. Se espera que, a través de ese foro, el alumnado intercambie experiencias y conocimientos, ayudándose mutuamente a resolver dichas dudas. También el profesorado de la asignatura intervendrá en las discusiones que pudieran suscitarse, si fuera necesario.

Bibliografía

[1] Chomsky Normal Form https://en.wikipedia.org/wiki/Chomsky_normal_form

[2] CYK algorithm https://en.wikipedia.org/wiki/CYK_algorithm

[3] Conversion of CFG to Chomsky Normal Form https://tinyurl.com/cnf-example

[4] Chomsky Normal Form https://courses.cs.washington.edu/courses/cse322/08au/lec14.pdf