Teoría de la Computación

Víctor Melchor

Clase 7

Teoría de Autómatas

Capítulo 21

Simplificación de GLC

Una gramática en la forma normal de Chomsky puede tener una estructura que facilite la prueba de propiedades. Pero, ¿qué hay de la implementación de un analizador para dicho lenguaje?

La nueva forma menos general aún no es muy eficiente de implementar. ¿Cómo podemos esperar un análisis mejor y más eficiente?

En una GLC podemos encontrar tres defectos que es necesario eliminar:

1. los factores comunes izquierdos
2. la recursividad por la izquierda
3. la ambigüedad

21.1 Factores Comunes Izquierdos

Una GLC G se dice que tiene factores comunes si hay por lo menos 2 reglas con el mismo símbolo en la parte izquierda y tienen algunos símbolos coincidentes en el prefijo de la parte derecha.

Se tendrá formalmente:

$$A ::= \delta\alpha_1|\delta\alpha_2|\cdots|\delta\alpha_n|\beta_1|\cdots|\beta_m \quad \text{con } n \geq 2, \ |\delta| > 0$$

Eliminación de Factores Comunes Izquierdos

Dada una GLC G con factores comunes izquierdos (FCI)

$$A ::= \delta\alpha_1|\delta\alpha_2|\cdots|\delta\alpha_n|\beta_1|\cdots|\beta_m \quad \text{con } n \geq 2,\ |\delta| > 0$$

Para eliminar los FCI realice la sustitución siguiente

Añadir un nuevo símbolo no terminal C de modo que:

$$A ::= \delta C|\beta_1|\cdots|\beta_m$$

$$C ::= \alpha_1|\alpha_2\cdots|\alpha_n$$

21.2 Recursividad por la Izquierda

Un símbolo no terminal A es recursivo por la izquierda si tiene una regla de la forma:

$$A \to Aw \qquad w \in \Sigma^*$$

Eliminación de Recursividad Izquierda

Las reglas de un símbolo no terminal A se pueden descomponer como:

$$\begin{cases} A & ::= A\alpha_1 | A\alpha_2 | \cdots | A\alpha_n \\ A & ::= \beta_1 | \beta_2 | \cdots | \beta_m \end{cases}$$

donde:

$\alpha_i, \beta_i \in \Sigma^*$

el primer símbolo de β_i es diferente de A

Podemos eliminar la recursividad por la izquierda introduciendo un símbolo no terminal Z de modo que:

$$A ::= \beta_1|\beta_2|\cdots|\beta_m|\beta_1 Z|\beta_2 Z|\cdots|\beta_m Z$$

$$Z ::= \alpha_1|\alpha_2|\cdots|\alpha_n|\alpha_1 Z|\alpha_2 Z|\cdots|\alpha_n Z$$

Lema: En una GLC cualquiera, una producción $A \to uBv$ se puede reemplazar por:

$$A \to uw_1v|uw_2v|\cdots|uw_nv$$

siendo $B \to w_1|w_2|\cdots|w_n$ todas las producciones de B

Ambigüedad

No hay algún algoritmo que nos permita eliminar la ambigüedad.

En el caso de los LLC que sólo tienen GLC ambigüas, es imposible eliminar la ambigüedad.

Sin embargo en algunos casos es posible resolver este problema analizando cuales son sus causas.

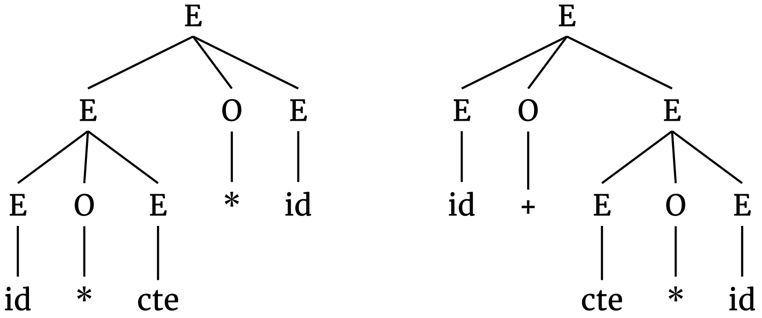
Ejemplo: Sea la gramática G para la definición de expresiones aritméticas, donde

$$\begin{aligned} \Sigma_T &= \{id, cte, (,), +, -, *, /\} \\ \Sigma_N &= \{E, O\} \\ S &= E \\ E &::= EOE|(E)|id|cte \\ O &::= +|-|*|/ \end{aligned}$$

Obtener el árbol de decisión para $w = id + cte * id$

Solución

Se obtiene dos derivaciones



Luego, G es ambigüa

Esto se debe a que no hay un orden de prioridad entre los operadores.

Para resolver esta ambigüedad consideremos:

1. la $*$ y $/$ tienen una prioridad más alta que + y $-$
2. si hay operaciones con la misma prioridad, se ejecutarán de izquierda a derecha.

Introduciremos los nuevos símbolos no terminales:

T término

A operador suma y resta

F factor

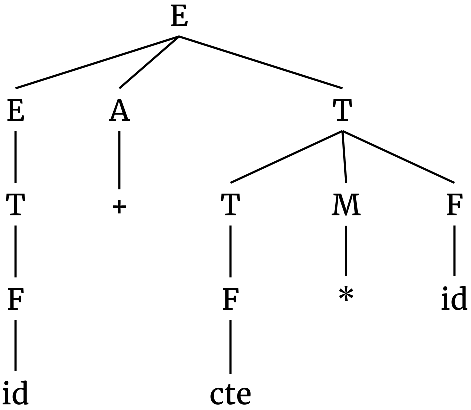
M operador multiplicación y división

y generamos la gramática equivalente G^2, donde:

$$\Sigma_N = \{E, T, F, A, M\}$$

$$
\begin{aligned}
S &= E \\
E &::= EAT|T \\
T &::= TMF|F \\
A &::= +- \\
M &::= *|/
\end{aligned}
$$

El árbol de derivación para la cadena $w = id + cte * id$



Esta gramática obliga que la multiplicación se realice antes que la suma.

21.3 Forma Normal de Greibach

Una GLC está en la Forma Normal de Greibach (FNG) si:

1. La variable inicial no es recursiva.
2. G no tiene variables inútiles
3. G no tiene producciones ε (salvo $S \to \varepsilon$ posiblemente)
4. Todas las reglas son de la forma:

$$
\begin{array}{ll}
A \to a & \text{(reglas simples)} \\
A \to aB_1B_2 \dots B_k & \text{donde } B_i \text{ son símbolos no terminales}
\end{array}
$$

Sheila Adele Greibach es una investigadora en lenguajes formales en computación,autómatas, teoría del compilador (en particular) y la informática.
Ella es una profesora emérito de Ciencias de la Computación en la Universidad de California, Los Ángeles.
Además de establecer la forma normal también investigó las propiedades de W-gramáticas , autómatas de pila , y problemas de decidibilidad.

En 1963, logró su doctorado en la Universidad de Harvard, aconsejada por Anthony Oettinger. El título de su tesis doctoral es "inversas de Generadores estructura de la frase".

Características

1. En cada paso de la derivación aparece un único terminal.
2. La derivación de una cadena de longitud n ($n \geq 1$) tiene exactamente n pasos

Teorema: Toda GLC G es equivalente a una gramática en FNG.

Método de Conversión

Para convertir una gramática a su FNG realice los siguientes pasos:

1. Enumere las variables en un orden arbitrario pero fijo en el procedimiento, donde S debe ser la variable de orden 1

2. Para cada variable A de la gramática original, de acuerdo al orden elegido, modifique las producciones de tal modo que el primer símbolo a la derecha de la flecha sea un terminal o una variable con mayor orden que el de A.

3. Utilice el Lema para modificar las producciones de las variables originales de tal modo que el primer símbolo a la derecha de la flecha sea un terminal.

   Se debe seguir el orden inverso de enumeración de las variables: última, penúltima, etc.

4. Utilizar de nuevo el Lema para modificar las producciones de las variables nuevas, de tal modo que el primer símbolo a la derecha de la flecha sea un terminal.

Ejemplo: Dada la gramática G

$$S ::= AA|a$$
$$A ::= AA|b$$

convertir a su FNG

Solución

Paso 1: S

A

Paso 2: Debemos eliminar la recursividad a la izquierda de la variable A. Introduciremos Z y reemplazamos la regla:

$$A ::= AA|b$$

por

$$\begin{cases} A ::= b|bZ \\ Z ::= A|AZ \end{cases}$$

Se obtiene la gramática:

$$\begin{cases} S ::= AA|a \\ A ::= b|bZ \\ Z ::= A|AZ \end{cases}$$

Paso 3: Reemplazando $A ::= b|bZ$ en la variable original S se obtiene

$$\begin{cases} S ::= bA|bZA|a \\ A ::= b|bZ \\ Z ::= A|AZ \end{cases}$$

Paso 4: Descomponemos las reglas de Z:

$$\begin{cases} Z ::= A \\ Z ::= AZ \end{cases}$$

usando $A ::= b|bZ$ Se obtiene finalmente

$$\begin{cases} S ::= bA|bZA|a \\ A ::= b|bZ \\ Z ::= b|bZ|bZZ \end{cases}$$

que ya está en FNG.

Ejemplo: Dada la gramática G

$$\begin{cases} S ::= AB|BC \\ A ::= AB|a \\ B ::= AA|CB|a \\ C ::= a|b \end{cases}$$

Convertir a su FNG

Solución

Paso 1: Ordenamos las variables

S

B

A

C

$$\begin{cases} S ::= AB|BC \\ B ::= AA|CB|a \\ A ::= AB|a \\ C ::= a|b \end{cases}$$

Paso 2: Eliminaremos la recursividad a la izquierda de A

$$\begin{cases} S ::= AB|BC \\ B ::= AA|CB|a \\ A ::= a|aZ \\ C ::= a|b \\ Z ::= B|BZ \end{cases}$$

Paso 3: Reemplazamos las variables originales para que el primer símbolo del cuerpo sea un terminal

1. Sustituimos $S := AB$ usando $A ::= a|aZ$

$$S ::= aB|aZB$$

2. Sustituimos $S ::= BC$ usando $B ::= AA|CB|a$ y $A ::= a|aZ,\ C ::= a|b$

$$S ::= AAC|CBC|aC$$
$$S ::= aAC|aZAC|aBC|bBC|aC$$

3. Sustituimos $B ::= AA$ usando $A ::= a|aZ$

$$B ::= aA|aZA$$

4. Sustituimos $B ::= CB$ usando $C ::= a|b$

$$B ::= aB|bB$$

se obtiene la gramática:

$$\begin{aligned}
S &::= aB|aZB|aAC|aZAC|aBC|bBC|aC \\
B &::= aA|aZA|aB|bB|a \\
A &::= a|aZ \\
C &::= a|b \\
Z &::= B|BZ
\end{aligned}$$

Paso 4: Reemplazamos en las reglas de las variables para que el primer símbolo del cuerpo sea un terminal. Se obtiene la gramática:

$$
\begin{aligned}
S &::= aB|aZB|aAC|aZAC|aBC|bBC|aC \\
B &::= aA|aZA|aB|bB|a \\
A &::= a|aZ \\
C &::= a|b \\
Z &::= aA|aZA|aB|bB|a|aAZ|aZAZ|aBZ|bBZ|aZ
\end{aligned}
$$

que ya está en FNG.