Cátedra Sintaxis y Semántica del Lenguaje

1 º CICLO DE CAPACITACION DOCENTE

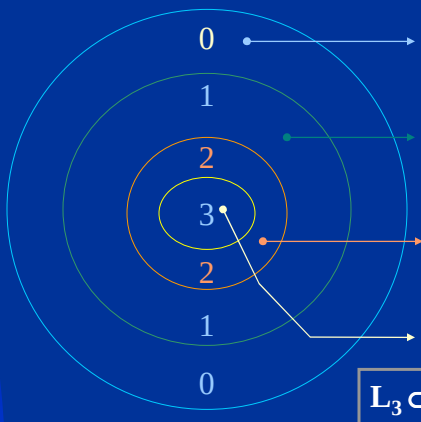
Simplificación de Gramáticas Tipo 2

- Forma Normal de Chomsky (FNC)
- Forma Normal de Greibach (FNG)

Agenda

- Jerarquía de Chomsky
- Gramáticas Tipo 2
- Sintaxis en los Leng. De Programación
- Aceptación de Cadenas vacías – Reglas λ
- Necesidad de Simplificación de una GT2
- Formas Normales
- Forma Normal de Chomsky
- Forma Normal de Graibach
- Resumen
- Bibliografía

La jerarquía de Chomsky



Estructurados por Frases o R.E.

Dependientes del Contexto

Independientes del Contexto

Regulares

$L_3 \subset L_2 \subset L_1 \subset L_0$

La jerarquía de Chomsky(Cont.)

Las restricciones que se le impongan a la conformación de las reglas de reescritura, dan origen a los distintos tipos de Gramáticas para la generación de lenguajes.

Nuestro tema de estudio son las gramáticas tipo 2, ya que son estas las que definen la mayoría de las Gramáticas de los lenguajes de Programación (Sintaxis)

Gramáticas Tipo 2

(según Libro de Cátedra - Giró)

Tipo 2: Gramáticas Independientes del Contexto

$$A \rightarrow \alpha$$

- donde: $A \in N$
- $\alpha \in (T \cup N)^*$

Parte Izquierda : Solo un Símbolo No Terminal
Parte Derecha : Sin Restricciones

Con esta definición es posible aceptar cadenas vacías

También es común definirlas sin reglas λ ó permitirlas sólo desde el axioma inicial

Sintaxis en los Lenguajes De Programación

• La Sintaxis en los Lenguajes de Programación es la forma en la que se escribirán los programas.

• Dar las reglas de sintaxis para un lenguaje significa indicar como se escriben las instrucciones, declaraciones y otras construcciones del lenguaje.

• El propósito prioritario de la Sintaxis es el de proporcionar una notación para comunicar la información entre el programador y el procesador de lenguaje (Compilador).

Aceptación de Cadenas vacias – Reglas λ

Este punto parece no ser Trivial:

• El no incluir reglas-λ no permitiría la generación de cadenas nulas (Lenguaje vacío).

• En Teoría de la Computación Brookshear, impone como condición en los teoremas de conversión desde GT2 a Formas normales de Chomsky y Greibach partir de los lenguajes independependientes de contexto que no contengan la cadena vacía. Y demuestra solo para la equivalencia para L – {λ}. Luego al parecer sin perdida de generalidad lo extiende para gramáticas con reglas -λ

Necesidad de Simplificación de una GT2

La Sintaxis de un Lenguaje de Programación es utilizada por:

- *Programadores (Usuarios)*
- *Implementadores (Desarrolladores)*

Análisis de las GT2 lo vamos a realizar desde dos perspectivas:

- *Poder de Representación*
- *Algoritmos eficientes para Análisis*

Poder de Representación

Desde : el Programador

Necesita saber exactamente cómo se debe escribir un programa, el formato de cada proposición, su puntuación, frases opcionales

Desde: *el Implementador*

Necesita saber el conjunto completo de programas bien escritos que debe aceptar el traductor

Como está definida las GT2 no presentan inconvenientes en ninguno de los casos

Eficiencia en los Algoritmos(1)

• En la Implementación de un compilador para un lenguaje de programación hay varias Etapas y Fases a cumplir.

• El análisis sintáctico no sólo sirve para controlar la estructura correcta de la frase, sino que prepara estructuras para usos posteriores.

• Estas estructuras servirán para el análisis semántico (si tiene sentido además de estar escrito en forma correcta) y posterior generación de instrucciones en el programa objeto.

Eficiencia en los Algoritmos(2)

Árbol de derivación Sintáctico

Este debe ser construido y se entrega como resultado del proceso de el análisis sintáctico y el mismo se verá afectado de acuerdo a como son impuestas las restricciones en la Gramática.

Partiendo de una Gramática Tipo 2 sin alterar el poder expresivo en la definición del lenguaje que describe, pueden obtenerse diversas formas equivalentes dispuestas en alguna forma especial " *Normalizada* " que llamaremos **formas Normales.**

Eficiencia en los Algoritmos(3)

Ejemplo 1

S → abcdefgS

S → abcdefg

Nos daría un Árbol sintáctico densamente tupido y casi incontrolable en los algoritmos de recorrido

Ejemplo 2

S → A ; A → B ; B → C

C → D ; D → a ; D → A

Nos daría un árbol sintáctico inútilmente profundo y delgado en donde los algoritmos de recorrido serian extensos e ineficientes.

Eficiencia en los Algoritmos(4)

Es deseable poder establecer las restricciones necesarias en las reglas de producción, de tal manera que los árboles de derivación sintácticos resultantes, no sean innecesariamente complejos o inútilmente sencillos.

Formas Normales

Existen dos modelos de formas normales. Las Formas Normales de Chomsky y de Greibach.

Para cada una de ellas vamos a determinar:

- Restricciones impuestas a las reglas de reescritura.
- Ventajas y desventajas de cada una.
- Forma de obtenerlas.
- Ejemplo práctico

Formas Normal de Chomsky (FNC)

Las reglas de producción pueden adoptar las siguientes formas:

$A := BC$

$A := a$

$S := \lambda$ Siendo S – Axioma Inicial

Donde: $A, B, C \in \Sigma N$ y $a \in \Sigma T$

Formas Normal de Chomsky (FNC)

Ventaja:

Todos los árboles de derivación son binarios y favorecen la etapa de generación de Código Intermedio.

Siempre es posible pasar a un código de 3 direcciones.

Desventaja:

Deja la posibilidad de permitir recursiones por la izquierda en uno o más pasos, que es un efecto no deseado en una gramática para la implementación de algoritmos.

Procedimiento de Conversión de una GT2 en FNC(1)

Para todas la producciones de la gramática, se debe realizar:

1) Si la producción está en FNC no se hace nada y se la deja como está

2) Si la parte derecha de la producción comienza con un símbolo terminal.

$$A: = a\beta \quad \text{con} \quad A \in \Sigma N$$
$$a \in \Sigma T$$
$$\beta \in (\Sigma T \cup \Sigma N)^{+}$$

Procedimiento de Conversión de una GT2 en FNC(2)

a) Se busca si existe alguna producción en donde el terminal en cuestión sea producido por un no terminal. O sea si existe un C: = a y sea la única producción de C

Si esto ocurre, a la producción original se la remplaza por:

$$A := C\beta$$

b) No existe ninguna producción C : = a , entonces se crea un nuevo símbolo terminal N, que N : = a , y nos queda:

$$N := a$$

$$A := N\beta$$

Procedimiento de Conversión de una GT2 en FNC(3)

3) La parte derecha de la producción comienza con un símbolo no terminal pero no continua con sólo un No terminal

$$A := B\eta$$

a) $\eta \in \Sigma T$ y $|\eta| = 1$

En este caso se busca si existe o se crea una producción con un símbolo no terminal que produzca el terminal deseado

$$A := Ba$$

$A := BN$ ya que existe N : = a

Procedimiento de Conversión de una GT2 en FNC(4)

b) $A := B\eta$ con $\eta \in (\Sigma T U \Sigma N)^+$

En este caso se crea una nueva regla con un nuevo símbolo no terminal

$$N' := \eta$$

Si es que esta regla no existiese, quedando

$$A := BN'$$

4) Se continua hasta que todas las producciones nuevas o existentes estén en FNC.

Ejemplo de Conversión de una GT2 en FNC(1)

G = ({1,2} , {A, B, C} , A, P)

P = {
A: = CB2 (producción 1)
A : = 1B (producción 2)
B : = BC (producción 3)
A : = λ (producción 4)
B : = 1 (producción 5)
C : = 2 } (producción 6)

Ejemplo de Conversión de una GT2 en FNC(2)

Tomo la producción 1

A : = CB2

La parte derecha comienza con un No terminal pero a continuación hay mas de un símbolo (No terminal o terminal) entonces creo un nuevo símbolo no terminal D que produce B2

D : = B2

A : = CD

La producción de A ha quedado en Forma Normal de Chomsky, pero la nueva D no.

Ejemplo de Conversión de una GT2 en FNC(3)

La parte derecha de D comienza con un no terminal pero está seguida de un terminal.

Ahora vemos que existe una producción C : = 2 (Producción 6), o sea que la producción D se puede rescribir.

FNC	D : = BC A : = CD	≡	A : = CB2

Ejemplo de Conversión de una GT2 en FNC(4)

Tomemos ahora la producción 2

A : = 1B

La parte derecha comienza con un símbolo terminal, así que hay que aplicar el paso “2b” ya que sólo debe haber una regla que llegue a 1

A : = EB

E : = 1

Las producciones 3, 4 ,5 6, ya está en FNC y no hay que hacerles nada.

Ejemplo de Conversión de una GT2 en FNC(5)

la nueva G´ en FNC queda :

G´ = ({1, 2} , {A, B, C, D, E}, A, P´)

P´ = { A : = CD
A : = EB
A : = λ
B : = BC
B : = 1
C : = 2
D : = BC
E : = 1 }

Formas Normal de Greibach (FNG)

Las reglas de producción pueden adoptar las siguientes formas:

$A := a\eta$

$S := \lambda$ Siendo S – Axioma Inicial

Donde: $A \varepsilon \Sigma N$; $a \varepsilon \Sigma T$; $\eta \varepsilon \Sigma N^*$

Formas Normal de Greibach (FNG)

Ventaja:

Resulta un representación sumamente útil para realizar el proceso de conversión de una GT2 a un autómata a Pila, que se derive en un analizador sintácticos LL.

Desventaja:

El árbol sintáctico resultante, puede no ser un árbol binario lo que en la Generación del Código intermedio necesitará de un proceso mas laborioso.

Procedimiento de Conversión de una GT2 en FNC(1)

1) Si la Gramática no está limpia debe limpiarse como primer paso.
2) Debe eliminarse la recursividad por izquierda.
3) No puede haber reglas que en su parte derecha comiencen por un símbolo No terminal.

Se establece un orden de los símbolos No terminales, y en las reglas que no están en FNG se producen los reemplazos necesarios, en el orden en que han sido fijados los símbolos No terminales.

4) Si todas las reglas están en FNG no se hace nada.

Ejemplo de Conversión de una GT2 en FNG(1)

G = ({1,2} , {A, B, C} , A, P)

P = { A: = CB2 (producción 1)
A : = 1B (producción 2)
B : = BC (producción 3)
A : = λ (producción 4)
B : = 1 (producción 5)
C : = 2 } (producción 6)

Ejemplo de Conversión de una GT2 en FNG(2)

1) La gramática ya está limpia.

No tiene reglas innecesarias

No tiene símbolos inaccesibles

No tiene símbolos superfluos Terminales y No terminales

Ejemplo de Conversión de una GT2 en FNG(3)

2) Eliminación de producciones recursivas por la izquierda

La producción 3 es recursiva **B := BC**

Para eliminarla se borran todas las producciones del símbolo B se crea un nuevo símbolo No Terminal **D** quedando.

B:= BC C → α

B:=1 1 → β ___

Creamos nuevas producciones para reemplazarlas en base al procedimiento establecido

P' = { B := βD ; B := β ; D := αD ; D := α }

Y nos queda

Ejemplo de Conversión de una GT2 en FNG(4)

B := 1D
B := 1
D := CD
D := C

Ahora eliminamos la regla de redenominación D := C y añadimos la regla D := 2 quedando

B := 1D
B := 1
D := CD
D := 2

Ejemplo de Conversión de una GT2 en FNG(5)

El Conjunto de reglas de Producciones hasta ahora nos queda:

P = { A := CB2
A := 1B
A := λ
B := 1D
B := 1
C := 2
D := CD
D := 2 }

Ejemplo de Conversión de una GT2 en FNG(6)

3) Se establece un orden de los símbolos No terminales resultando - **A B D C** –

* Tomamos A := CB2

  y generamos A:= 2B2 ya que C := 2

* Tomamos D := CD

  y generamos D := 2D ya que C := 2

Y el conjunto de producciones nos queda

Ejemplo de Conversión de una GT2 en FNG(7)

El Conjunto de reglas de Producciones hasta ahora nos queda:

P = { A := 2B2
A := 1B
A := λ
B := 1D
B := 1
C := 2
D := 2D
D := 2 }

Ejemplo de Conversión de una GT2 en FNG(8)

4) nos queda solamente una producción que no está en FNG

A := 2B2

Como existe una producción C := 2 y sólo existe una sola producción C, no es necesario crear otro símbolo adicional y la podemos reemplazar, quedando

A := 2BC

Finalmente la gramática en FNG queda.

Ejemplo de Conversión de una GT2 en FNG(9)

G = ({1,2} , {A, B, C,D} , A, P)

P = { A := 2BC
A := 1B
A := λ
B := 1D
B := 1
C := 2
D := 2D
D := 2 }

Resumen(1)

La normalización de las GT2 con formas normales ya sea la de Chomsky o la de Greibach, es sumamente útil en el proceso de desarrollo de un Compilador, no sólo en la fase de análisis sintáctico, sino que tiene un alto impacto en las fases posteriores.

Las restricciones impuestas a las reglas de producción no alteran el poder de generación de la gramática, sino que el impacto se evidencia en cómo se construirá el árbol de derivación sintáctico.

Este es el producto de salida de la fase de análisis sintáctico y es utilizado por todas las fases posteriores.

Resumen(2)

En la etapa de síntesis sería mas deseable partir de un árbol sintáctico generado a partir de una FNC, ya que tengo un árbol binario y por lo tanto el algoritmo para convertir a código de tres direcciones resulta directo.

Si tengo normalizada la gramática en FNG la ventaja significativa la obtengo directamente en la etapa de análisis sintáctico ya que se facilitan los algoritmos de los ***Analizadores Sintácticos*** LL

Bibliografía(1)

- **Introducción a la Informática Teórica**
  J. Giró, Cuidad Gráfica
- **Teoría de la Computación**
  J. G. Brookshear, Addison-Wesley Iberoamericana
- **Lenguajes, Gramáticas y Autómatas**
  P. Isasi / P. Martínez / D. Borrajo, Addison-Wesley
- **Introducción a la Teoría de Autómatas Lenguajes y Computación**
  J. E. Hopcroft / J.D. Ullman, Addison-Wesley P. C.
- **Fundamentos de Compiladores**
  Karen A. Lemone CECSA

Bibliografía(2)

- **Compiladores: principios, técnicas y herramientas**
  A. V. Aho / R. Sethi / J. D. Ullman, Addison-Wesley
- **Lenguajes de Programación – Diseño e implementación**
  Terrence W. Pratt, Prentice Halll
- **Teoría de Autómatas y Lenguajes Formales**
  Dean Kelley, Prentice Hall