Lenguajes Formales

GRAMÁTICAS INDEPENDIENTES DEL CONTEXTO

Facultad de Ingeniería – UNJu

Ing. Fabiana R. Aragón

Gramáticas Independientes del Contexto (GIC)

Este tipo de gramáticas y autómatas son importantes desde el punto de vista de la informática:

- Permiten la descripción de la mayoría de los lenguajes de programación.
- Permiten la construcción de compiladores.

Gramáticas Independientes del Contexto (GIC)

Las técnicas para preparar una gramática a efectos de ser tratada eficientemente por un autómata que reconozca el lenguaje generado por ella son:

- Gramática Limpia
- Gramática Bien Formada
- Forma Normal de Chomsky
- Forma Normal de Greibach

Gramática Limpia

Dada una gramática de tipo 2 se definen:

1. Reglas innecesarias:

   Tienen la forma A::=A ($A \in \Sigma_N$).

   Estas reglas se deben eliminar de las gramáticas ya que no generan derivaciones útiles.

2. Símbolos inaccesibles:

   Son símbolos no terminales $A \in \Sigma_N$ que **no** pueden ser alcanzados desde el axioma de la gramática, S.

   Es decir, no hay derivación $S \rightarrow xAy$

Gramática Limpia

Ejemplo: Dada la siguiente gramática.

G_1=({0,1,2,3}, {A,B,C,D,E}, A, P)

P={(A::=D0), (A::=E10), (A::=λ), (B::=1C3), (C::=C), (D::= 1A), (E::=1E)}

La regla (C::=C) es innecesaria y los símbolos B y C son inaccesibles.

G'_1=({0,1,2,3}, {A,D,E}, A, P)

P'={(A::=D0), (A::=E10), (A::=λ), (D::= 1A), (E::=1E)}

Gramática Limpia

3. Símbolos Superfluos: dependiendo del conjunto al que pertenecen tenemos:

- **Símbolo No Terminal superfluo, A:** es aquel del que solo se pueden derivar palabras en las que existe al menos un símbolo no terminal, o, lo que es lo mismo, $A \not\rightarrow^* x$ $(x \in \Sigma_T^*)$.

- **Símbolo terminal superfluo, a:** es aquel que no puede ser alcanzado por derivación desde el axioma; es decir, no existe ninguna producción $(A ::= \alpha a \beta \in P)$

Gramática Limpia

Procedimiento de eliminación de los símbolos superfluos de una gramática.

I. Se marcan los símbolos no terminales que estén en la parte izquierda de una producción y en cuya parte derecha solo aparezcan símbolos terminales o λ.

II. Sucesivamente, se van marcando los símbolos no terminales que aparezcan en la parte izquierda de producciones en las que solo haya combinaciones de símbolos terminales λ, o símbolos no terminales marcados.

III. Por ultimo se eliminan aquellos símbolos terminales que no aparezcan en la parte derecha de ninguna regla y todas las producciones de los símbolos no terminales no marcados.

Gramática Limpia

Ejemplo: en la siguiente gramática

$G'_1=(\{0, 1, 2, 3\}, \{A, D, E\}, A, P)$

P={(A::=D0), (A::=E10), (A::=λ), (D::= 1A), (E::=1E)}

Eliminación de Símbolos Superfluos:

Buscamos No Terminales superfluos.

P:

A ::= D0

A ::= E10

A ::= λ

D ::= 1A

E ::= 1E

Gramática Limpia

$G'_1=(\{0, 1, 2, 3\}, \{A, D, E\}, A, P)$

$P=\{(A::=D0), (A::=E10), (A::=\lambda), (D::= 1A), (E::=1E)\}$

A ::= D0

A ::= E10

A ::= λ

D ::= 1A

E ::= 1E

E es símbolo No Terminal superfluo

Se eliminan las reglas donde aparecen los No Terminales superfluos.

Gramática Limpia

G_1' =({0, 1, ~~2, 3~~}, {A, D}, A, P)

P={(A::=D0), (A::=λ), (D::= 1A)}

2 y 3 serían Terminales superfluos

Se eliminan aquellos símbolos terminales que no aparezcan en la parte derecha de ninguna regla (símbolos Terminales superfluos).

G_1''=({0, 1}, {A, D}, A, P)

P={(A::=D0), (A::=λ), (D::= 1A)}

Gramática Limpia

Definición:

Una gramática está limpia si no tiene reglas innecesarias, símbolos inaccesibles, ni símbolos superfluos.

$G_1'' = (\{0,1\}, \{A,D\}, A, P)$

P:

A::=D0

A::=λ

D::= 1A

La gramática G_1'' esta limpia

Gramática Bien Formada

4. Reglas No generativas: una regla es no generativa cuando

$$A ::= \lambda,\ A \neq S$$

Ejemplo:

$G_2 = (\{0\}, \{A, B, C\}, A, P)$

P={(A::=C0B), (A::= λ), (B::=BC), (B::= λ), (C::=0B), (C::= λ)}

Gramática Bien Formada

Eliminación de Reglas No Generativas:

$G_2=(\{0\}, \{A, B, C\}, A, P)$

$P=\{(A ::= C0B), (A ::= \lambda), (B ::= BC), (B ::= \lambda), (C ::= 0B), (C ::= \lambda)\}$

Eliminación de la regla $B ::= \lambda$

$A ::= C0B$	$B ::= BC$	$C ::= 0B$
$A ::= C0$	$B ::= C$	$C ::= 0$
$A ::= \lambda$		$C ::= \lambda$

Gramática Bien Formada

Eliminación de Reglas No Generativas (cont.):

G_2'= ({0}, {A, B, C}, A, P)

P={(A::=C0B), (A::=C0), (A::= λ), (B::=BC), (B::=C), (C::=0B), (C::=0), (C::= λ)}

Eliminación de la regla C::= λ

A ::= C0B
A ::= 0B
A ::= C0
A ::= 0
A ::= λ

B ::= BC
B ::= B
B ::= C
B ::= λ

C ::= 0B
C ::= 0

En este paso se generaron las reglas: B::=B y B::= λ

Gramática Bien Formada

Eliminación de Reglas No Generativas (cont.):

G_2'= ({0}, {A, B, C}, A, P)

P={(A::=C0B), (A::=0B), (A::=C0), (A::=0), (A::= λ), (B::=BC), (B::=B), (B::=C) (B::= λ), (C::=0B), (C::=0)}

Se elimina B ::= B por ser regla innecesaria.

Se elimina nuevamente la regla B::= λ

A ::= C0B
A ::= 0B
A ::= C0
A ::= 0
A ::= λ

B ::= BC
B ::= C

C ::= 0B
C ::= 0

Gramática Bien Formada

La gramática resultante es:

$G_2'=(\{0\}, \{A, B, C\}, A, P)$

P={(A::=C0B), (A::=0B), (A::=C0), (A::=0), (A::= λ),

(B::=BC), (B::=C), (C::=0B), (C::=0)}

Gramática Bien Formada

5. Reglas de redenominación: son reglas de la forma

$$A::=B \text{ con } A, B \in \Sigma_N$$

Ejemplo:

$G_2'=(\{0\}, \{A, B, C\}, A, P)$

P={(A::=C0B), (A::=0B), (A::=C0), (A::=0), (A::= λ), (B::=BC), (B::=C), (C::=0B), (C::=0)}

Para eliminarlas, se borra esa regla y se genera una nueva producción A::=x por cada regla B::=x, $x \in \Sigma^*$

Gramática Bien Formada

Eliminación de reglas de redenominación:

$G_2'=(\{0\}, \{A, B, C\}, A, P)$

P={(A::=C0B), (A::=0B), (A::=C0), (A::=0), (A::= λ), (B::=BC), (B::=C),

(C::=0B), (C::=0)}

En la regla B::=C, se reemplaza C por las partes derechas de las producciones de C.

$G_2'' =(\{0\}, \{A, B, C\}, A,P)$

P={(A::=C0B), (A::=0B), (A::=C0), (A::=0), (A::= λ), (B::=BC), (B::=0B), (B::=0),

(C::=0B), (C::=0)}

Gramática Bien Formada

Definición:

Una gramática está bien formada si está limpia, no tiene reglas no generativas y no tiene reglas de redenominación.

$G_2'' = (\{0\}, \{A, B, C\}, A, P)$

P={(A::=C0B), (A::=0B), (A::=C0), (A::=0), (A::= λ), (B::=BC), (B::=0B), (B::=0), (C::=0B), (C::=0)}

La gramática G_2'' está bien formada

Forma Normal de Chomsky (FNC)

Las gramáticas de tipo 2, se pueden transformar en gramáticas equivalentes expresadas en la Forma Normal de Chomsky.

En esta forma de representar las gramáticas, las producciones pueden tener las siguientes formas:

$$P=\{(A::=BC) \text{ ó } (S::=\lambda) \text{ ó } (A::=a) / A, B, C \in \Sigma_N, a \in \Sigma_T\}$$

FNC

Los árboles de derivación, salvo en las derivaciones correspondientes a las hojas, son binarios.

Se debe partir de una gramática bien formada.

El algoritmo para transformar una gramática en su FNC equivalente a cada producción P de la gramática, reasigna el nuevo valor del conjunto de producción P’ a lo que le devuelve la función FNC.

Esta función trata de convertir todas las producciones que no están el en formato FNC a dicho formato.

FNC

Los casos que se pueden presentar son los siguientes:

- La producción ya está en FNC. No se hace nada.

- La parte derecha de producción comienza por un símbolo terminal **a**. Ejemplo: (**B** ::= **aD**)

Lo que se hace es buscar si existe una producción de un símbolo No Terminal que sólo vaya al símbolo **a** (**C** ::= **a**); es decir, que no haya más producciones en las que **C** aparezca en la parte izquierda de la producción.

En caso de que se cumpla lo anterior, se puede sustituir en la producción inicial el símbolo **a** por el **C** y seguir tratando el resto de la producción.

Si no se encuentra ningún símbolo **C** que cumpla la premisa, se crea un nuevo símbolo **N**, se crea una nueva regla (**N** ::= **a**), y se sustituye **a** por **N** en la producción original.

FNC

- La parte derecha de la producción comienza por un símbolo no terminal ***B***, pero tiene mas de dos símbolos (supongamos que es ***By***). Ejemplo: (**D::= By**)

  Por medio de la función FNC-auxiliar, puede haber dos casos:

  1) ***y*** $\in \Sigma_T$: En este caso se sustituye ***y***, en esa producción, por un símbolo No Terminal que solo vaya a ***y*** (existente o nuevo).

  2) ***y*** $\in \Sigma_N$: En este caso, se crea un nuevo símbolo No Terminal N', se crea una nueva regla que tenga en la parte izquierda a N' y en la parte derecha a ***y***, se sustituye ***y*** por **N'** en la producción original y se vuelve a tratar la nueva regla formada **N::=y**.

FNC - Ejemplo

Formas FNC:
(A::=BC) ó (S::=λ) ó (A::=a)

G_3=({0, 1, 2}, {A, B, C}, A, P)

P={(A::=CB2), (A::=1B), (A::= **λ**), (B::=BC), (B::=1), (C::=2)}

A::=CB2 no está en FNC.

B2 $\notin \Sigma_T$, por lo que se crea un nuevo no terminal D. Se crea D::=B2, se crea A::=CD que está en FNC, y se elimina la regla A::=CB2.

Como D::=B2 no está en FNC debemos tratarla. Como existe C::=2, se añade la regla D::=BC que está en FNC, y se elimina D::=B2.

A::=1B no esta en FNC. No hay un símbolo No Terminal que sólo derive a 1 (distinto de B ya que no podemos tener una regla del tipo A::=BB), se crea uno nuevo E::=1, se añade la regla A::=EB, y se elimina la regla A::=1B. Tanto E::=1 como A::=EB están en FNC.

A::=λ, ya esta en FNC

B::=BC, ya está en FNC

B::=1, ya esta en FNC

C::=2, ya está en FNC

FNC - Ejemplo

La gramática final resultante es:

G_3'= ({0, 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)}