Tema 4

Lenguajes Formales

Universidad Europea
LAUREATE INTERNATIONAL UNIVERSITIES

Autor: Prof. Leopoldo Santos Santos

2021

Lenguajes Formales. Indice

- ¿Cuál fue la necesidad de los lenguajes formales?
- Definiciones básicas.
- Operaciones con palabras.
- Operaciones con lenguajes.
- Gramáticas formales.
- Derivaciones y árboles de derivación.
- Ambigüedad.
- Eliminación de la ambigüedad.

¿Cuál fue la necesidad de los lenguajes formales?

- Allá por 1954, se inició el desarrollo del **lenguaje de programación FORTRAN** que, aunque fue un verdadero compilador, adolecía de una **gran complejidad.**
- **Noam Chomsky** empezó con el estudio del **lenguaje natural,** con la idea de estructurarlo, y consiguió una revolución en este campo.

Gramáticas Tipo 0	**Sin restricciones.** Estas gramáticas tienen que tener en su parte izquierda al menos un símbolo no terminal.
Gramáticas Tipo 1	**Dependientes del contexto**. Se las denomina **dependientes del contexto** porque hay que tener en cuenta los símbolos que vienen antes y después del que queremos sustituir (su contexto).
Gramática Tipo 2	**Independientes del contexto. Generan lenguajes independientes del contexto y se caracterizan porque en la parte izquierda de una producción solo pueden tener un símbolo no terminal.**
Gramática Tipo 3	**Expresiones regulares**. Estas **son las gramáticas más restrictivas** y generan lenguajes regulares. En su parte izquierda tienen solo un **no terminal** y **en su parte derecha tienen solo un terminal**.

Definiciones Básicas

- **Carácter o símbolo**: Es el componente indivisible y elemental con el que se compone un texto.
  - Ejemplos: a, b, 5, [,)
- **Alfabeto Σ**: Es un conjunto de símbolos. Para los lenguajes formales, utilizamos alfabetos que contienen una cantidad finita de símbolos. Ejemplos:
  - $\sum_1$ = (a, b, c, d, e , f, g, h, i).
  - $\sum_2$ = (0, 1, 2, 3, 4, 5, 6, 7 ,8, 9).
  - $\sum_3$ = El alfabeto del lenguaje de programación C.
- **Palabra *w:*** Secuencia finita de letras del alfabeto
  - abc → es una palabra definida sobre $\sum_1$
  - 12 → es una palabra definida sobre $\sum_2$
  - main → es una palabra definida sobre $\sum_3$

Definiciones Básicas

- **Palabra vacía λ, ε**: Es una palabra que no contiene símbolos (es una convención similar al número 0 en matemáticas). Palabra de longitud 0
- **Universo $W(\Sigma)$:** Lo constituyen todas las palabras que pueden formarse con símbolos del alfabeto ∑, incluyendo a la palabra vacía (λ). Contiene, por tanto, un número infinito de palabras
  - Supongamos el alfabeto $\sum_1$ = (a, b, c, d, e , f, g, h, i). El universo de ese alfabeto sería: W($\sum_1$) = {λ, a, aa, ba, ab, ac, ca, de, …}
- **Lenguaje $L(\Sigma)$:** Es cualquier subconjunto del universo de ese alfabeto, y por tanto, puede ser también infinito.
  - Supongamos el alfabeto $\sum_1$ = (a, b, c, d, e , f, g, h, i). Posibles lenguajes de ese alfabeto podrían ser:
    - $L_1\sum_1$ = {λ, aa, bb, cc, dd}
    - $L_2\sum_1$ ={ba, ca, da, fa, ga, ha}

Operaciones con Palabras

- **CONCATENACIÓN, POTENCIA Y REFLEXION**

- **Concatenación**: Dadas dos palabras, α y β, de un mismo alfabeto ∑, la concatenación de estas dos palabras es una nueva palabra formada por los símbolos de α, seguido de los símbolos de β. Esta operación se representa mediante un punto α.β, que suele omitirse.
- Ejemplo: Sea $\sum_1$ = (a, b, c, d, e , f, g, h, i), si α = ba y β = ca, = baca
  - Esta operación no es conmutativa puesto que β.α, da otra palabra diferente, en este caso sería, β.α = caba
  - El elemento neutro de esta operación es la palabra vacía (λ).

Operaciones con Palabras

- **Potencia**: Si concatenamos la misma palabra varias veces efectuamos la operación potencia. Esto sería la potencia i-ésima de esa palabra.
  - Ejemplo: Sea $\sum_1$ = (a, b, c, d, e , f, g, h, i), si α = ba, α^2 = α.α = baba
    - Por definición, cualquier palabra elevada a 0 es la palabra vacía, $\alpha^0 = \lambda$

- **Reflexión**: La palabra inversa de una dada, se construye invirtiendo el orden de los símbolos de la palabra.
  - Ejemplo: Sea $\sum_1$ = (a, b, c, d, e , f, g, h, i), si α = ba, α^{-1} = ab

Operaciones con Lenguajes

- Las operaciones que nos interesan sobre un lenguaje son: **Unión**, **Concatenación**, **Clausura Positiva** y **la Clausura o (Cierre de Kleene)**. Hay otras operaciones como la Potencia, Reflexión, Resta o Complemento que no las vamos a necesitar.
  - A la operación de **Cierre** también se le denomina **Clausura**

- **Unión:** Sean L_1 y L_2 dos lenguajes sobre un alfabeto $\sum$, L_1 U L_2 es el lenguaje formado por las palabras de L_1 y por las palabras de L_2.
  - Ejemplo: Supongamos el alfabeto $\sum_1$ = (a, b, c, d, e , f, g, h, i) y dos lenguajes de ese alfabeto podrían ser: $\mathbf{L_1 \sum_1}$ = {λ, aa, bb, cc, dd} y $\mathbf{L_2 \sum_1}$ ={ba, ca, da, fa, ga, ha}. La unión de L_1 con L_2 sería:
    - $\mathbf{L_1}$ **U** $\mathbf{L_2}$ = { λ, aa, bb, cc, dd, ba, ca, da, fa, ga, ha}, que cumple con la propiedad conmutativa, es decir que L_2 U L_1 daría el mismo resultado.

Operaciones con Lenguajes

- **Concatenación:** Sean L_1 y L_2 dos lenguajes sobre un alfabeto $\sum$, L_1 . L_2 es el lenguaje formado por las palabras de L_1 concatenado con las palabras de L_2.
  - Ejemplo: A partir del alfabeto y los lenguajes descritos en el párrafo anterior, $\mathbf{L_1 . L_2}$ = { ba, ca, da, fa, ga, ha, aaba, aaca, aada,…}.
- **Cierre Positivo ($\sum^+$):** Es el lenguaje formado por todas las palabras que pueden formarse con símbolos del alfabeto $\sum$, **excluyendo la palabra vacía**.
  - Si $L_2\sum_1$ ={ba, ca, da, fa, ga, ha}, el cierre positivo de ese lenguaje será $L_2(\sum_1)^+$ = {ba, baca, bada, ca, caba, cada, cafa, …}
- **Cierre o cerradura de Kleene ($(\sum^*)$):**Es el lenguaje formado por todas las palabras que pueden formarse con símbolos del alfabeto $\sum$, **incluyendo la palabra vacía**, que siempre será parte del cierre.
  - Si $L_2\sum_1$ ={ba, ca, da, fa, ga, ha}, el cierre de ese lenguaje será $L_2(\sum_1)^*$ **=** {**λ**, ba, baca, bada, ca, caba, cada, cafa, …}.

Gramáticas Formales

- **Una gramática formal** es una descripción de la estructura de las sentencias que forman un lenguaje, entendiendo por lenguaje, en este contexto, **un lenguaje de programación**.

- La forma en la que se describe una gramática (G) es mediante cuatro términos:
  - El alfabeto de los símbolos terminales ($\sum_T$),
  - El alfabeto de los símbolos no terminales ($\sum_N$),
  - El axioma o símbolo inicial de la gramática (**S**)
  - Un conjunto de producciones (**P**).

- Se denota de la siguiente forma:

$$\mathbf{G = \{\sum_T, \sum_N, S, P\}}$$

Gramáticas Formales

- **Símbolos terminales ($\sum_T$)**: Representa al conjunto de palabras reservadas de un lenguaje de programación. Esto incluye los caracteres especiales y los operadores tanto aritméticos como lógicos. Ejemplo: En el lenguaje de programación C:
  - $\sum_T$ = {main, {, }, #, include, if, then, define, int, float, char, double, ;, +, -, *, >, >=, |, ...}
  - Se denotan con letras minúsculas: Ej: **int, float, if**

- **Símbolos no terminales ($\sum_N$)**: Representa el conjunto de variables que utilizamos en las producciones de las gramáticas como símbolos de transición. Esto incluye el símbolo inicial de la gramática (S) y las variables que utilicemos.
  - Se denota con letras mayúsculas.

Gramáticas Formales

- **Símbolo inicial (S)**: Es un símbolo no terminal a partir del cual se obtienen todas las palabras del lenguaje y será el primer símbolo de la gramática G. También se le denomina **el *axioma*** de la gramática.

- **Conjunto de producciones (P):** Es un conjunto de reglas que indica como se obtienen las palabras del lenguaje definido por G. Establece las relaciones entre los símbolos terminales y los no terminales.
- Para definirlas se utiliza la **Forma de Backus-Naur (BNF)**. Su origen está en la especificación del lenguaje de programación ALGOL60, a mediados de los 60. La BNF es un metalenguaje, es decir, un lenguaje con el que se describen otros lenguajes y se describe de acuerdo con la siguiente tabla:

Gramáticas Formales

- **Conjunto de producciones (P):**

Símbolo	Significado
→ ::=	"Se define como". Se utilizan los dos símbolos indistintamente
|	alternativa
[a]	Una o ninguna ocurrencia de a. Expresan opcionalidad
{a}	Número arbitrario de ocurrencias de a (cero o mas repeticiones)

E → E + E
E → E * E
E → - E
E → (E)
E → id

Gramáticas Formales

Gramáticas Tipo 0	**Sin restricciones.** Estas gramáticas tienen que tener en su parte izquierda al menos un símbolo no terminal.
Gramáticas Tipo 1	**Dependientes del contexto**. Se las denomina **dependientes del contexto** porque hay que tener en cuenta los símbolos que vienen antes y después del que queremos sustituir (su contexto).
Gramáticas Tipo 2	**Independientes del contexto. Generan lenguajes independientes del contexto y se caracterizan porque en la parte izquierda de una producción solo pueden tener un símbolo no terminal.**
Gramáticas Tipo 3	**Expresiones regulares**. Estas **son las gramáticas más restrictivas** y generan lenguajes regulares. En su parte izquierda tienen solo un **no terminal** y **en su parte derecha tienen solo un terminal**.

Derivaciones y Árboles de Derivación

- Se trata de ver como, a partir del símbolo inicial de la gramática se puede derivar una secuencia de símbolos utilizando las producciones de esa gramática y esto se hace mediante derivaciones.
- Estas derivaciones se representan graficamente mediante un **árbol de derivación** o también llamado **Árbol de Análisis Sintáctico**.

  – ¿Como lo vemos?

Gramática	**Sentencia**
E → E + E E → E * E E → - E E → (E) E → id	(id * id)

Ambigüedad

- **La ambigüedad** se produce cuando hay mas de una manera de reconocer una palabra o sentencia a partir del símbolo inicial de la gramática.
- La ambigüedad se puede dar a varios niveles: sentencia, gramática o lenguaje.
  - Una sentencia es ambigua si existe mas de una derivación para reconocerla en una gramática.
  - Una gramática es ambigua si existe en su lenguaje una sentencia ambigua, es decir, es posible derivar una sentencia por mas de una derivación.
  - Un lenguaje es ambiguo si existe una gramática ambigua que lo genera. Si todas las gramáticas que lo generan son ambiguas, entonces se denomina al lenguaje inherentemente ambiguo.

Ambigüedad

- **Recursividad por la izquierda:** Se produce este tipo de recusividad cuando el primer símbolo no terminal aparece tanto en la parte derecha como en la izquierda de una producción.
  - Ejemplo: **A** → **A**α, donde α representa cualquier combinación de terminales y no terminales (no tiene porque ser un solo símbolo). Veremos mas adelante como se elimina esta recursividad.
- **Recursividad por la derecha:** Si el símbolo no terminal que aparece el último en la parte derecha de la producción es igual al de la parte izquierda.
  - Ejemplo: A → αA
- **Factorizar por la izquierda**: Esta solución se aplica cuando hay varias alternativas en una sentencia que comienzan igual . Ejemplo: A → Bα | Bx, tenemos que eliminar esta ambigüedad reescribiendo las producciones

Eliminar la ambigüedad

- **Eliminar la recursividad a izquierdas**: Partimos de la siguiente producción A → Aα | β, donde tanto α como β representan conjuntos de terminales y no terminales. La regla a aplicar es la siguiente:

  A → β A´

  A´→ α A | λ

  – Ejemplo: E → E + T | T, donde α = + T y β = T. Por tanto el resultado sería:

    E → T E´

    E´→ + TE | λ

Eliminar la ambigüedad

- **Factorizar por la izquierda:** Puesto que parte de alternativas de una producción que comienzan igual, tenemos que determinar la parte común mas larga entre estas alternativas y a partir de ahí se sustituye por un no terminal que actuará de discriminante. Si tenemos $A \rightarrow \alpha \beta_1 | \alpha \beta_2 | \ldots | \alpha \beta_n | \theta$, donde θ representa otras alternativas que no comienzan con α, se hace lo siguiente:

  $A \rightarrow \alpha A' | \theta$

  $A' \rightarrow \beta_1 | \beta_2 | \ldots | \beta_n$

  - Ejemplo: $E \rightarrow S\ d\ E | S$ donde la parte común, α, es igual a S, por tanto aplicando la regla anterior:

    $E \rightarrow S A'$

    $A' \rightarrow d E | \lambda$