Universidad de Huelva

Grado en Ingeniería Informática

Tema 4: Gramáticas y Lenguajes Formales. La jerarquía de Chomsky.

Resumen

Autor: Alberto Fernández Merchán
Asignatura: Algorítmica y Modelos de Computación

2021

1. Introducción

Se establecen las siguientes relaciones entre los tipos de gramáticas que estableció Chomsky y la clase de lenguajes reconocibles por ciertas máquinas:

Gramática	Lenguajes	Máquinas
Tipo 0	Sin restricciones	Máquinas de Turing
Tipo 1	Sensible al contexto	Autómatas linealmente acotados
Tipo 2	Libre de contexto	Autómatas de pila
Tipo 3	Regular	Autómatas Finitos

2. Gramáticas y Lenguajes Formales

2.1. Lenguajes Formales

2.1.1. Alfabeto

Se llama alfabeto a un conjunto finito, no vacío, cuyos elementos se denominan símbolos. Se definen mediante la enumeración de los símbolos que contienen.
Se usarán letras griegas mayúsculas para representar un alfabeto.

2.1.2. Palabra

Una palabra es una secuencia finita de símbolos de un alfabeto.
Se usarán letras minúsculas para representar las palabras de un alfabeto.

Longitud de una palabra: Número de símbolos que la componen.

Palabra vacía: Palabra con longitud cero. Se representa mediante λ.

Lenguaje Universal: Se define sobre un alfabeto Σ ($W(\Sigma)$). Es el conjunto de palabras que se pueden formar con las letras del alfabeto. Es un conjunto infinito.
Por ejemplo: $A = \{a\}$ El alfabeto A tiene solo 1 símbolo.
Por tanto, $W(A) = \{\lambda, a, aa, aaa, ...\}$
La palabra vacía pertenece a todos los lenguajes universales.

2.1.3. Operaciones con palabras

- Concatenación de palabras:

  Sean dos palabras x e y tales que $x, y \in W(\Sigma)$ y que la longitud de las palabras es $|x| = i, |y| = j$.
  Se llama concatenación de las palabras x e y a otra palabra z obtenida tras colocar las letras de y detrás de las letras de x.
  Se cumple que $|z| = |x| + |y|$.

  - Es una **operación cerrada**. Concatenar dos palabras de $W(A)$ resulta en otra palabra del mismo lenguaje universal.
  - Cumple la propiedad de **asociatividad**: $x(yz) = (xy)z$.
  - El **elemento neutro** es la palabra vacía (λ).
    Se cumple que: $\lambda x = x\lambda = x$

- No cumple la propiedad **conmutativa**. $xy \neq yx$.

- Potencia de una palabra:

  Se denomina potencia i-ésima de una palabra a la concatenación consigo misma. Se cumple que:

  - $x^i = xxxxx...x$ (i veces).
  - $x^{i+1} = x^i x \ (i > 0)$
  - $x^i x^j = x^{i+j} (i, j > 0)$
  - $x^0 = \lambda$
  - $|x^i| = i \cdot |x|$

- Reflexión de palabras

  Siendo $x = A_0 A_1 ... A_n$, se denomina palabra inversa de x a:
  $x^{-1} = A_n A_{n-1} ... A_0$

2.1.4. Lenguajes

Un lenguaje sobre el alfabeto Σ es cualquier subconjunto de $W(\Sigma)$.
El conjunto vacío $\varnothing$ es un subconjunto de $W(\Sigma)$. $\varnothing \neq \{\lambda\}$.

Estos dos conjuntos serán lenguajes sobre cualquier alfabeto. El alfabeto en sí también es considerado un lenguaje(formado por todas las posibles palabras de una letra).

2.1.5. Operaciones con Lenguajes

- Unión de lenguajes: Considerando dos lenguajes diferentes definidos sobre el mismo alfabeto $L1, L2 \subset W(\Sigma)$.

  Lenguaje formado por las palabras de ambos lenguajes.
  $L1 \cup L2 = \{x : x \in L1 \vee L2\}$

  - Operación Cerrada: El resultado estará definido en el mismo alfabeto.
  - Propiedad Asociativa: $(L1 \cup L2) \cup L3 = L1 \cup (L2 \cup L3)$
  - Elemento neutro: $L \cup \varnothing = L$
  - Propiedad conmutativa: $L1 \cup L2 = L2 \cup L1$
  - Propiedad de idempotencia: $L1 \cup L1 = L1$

- Concatenación de lenguajes:

  Considerando dos lenguajes definidos sobre el mismo alfabeto, la concatenación de estos lenguajes será:
  $L1 \bullet L2 = \{xy : x \in L1 \wedge y \in L2\}$
  Las palabras de este lenguaje estarán formadas por la concatenación de cada palabra del primero con otra del segundo.

  - Concatenación con el lenguaje vacío: $\emptyset L = \emptyset$
  - Operación cerrada: Resulta en un lenguaje sobre el mismo alfabeto.
  - Propiedad asociativa: $(L1\ L2)\ L3 = L1\ (L2\ L3)$
  - Elemento neutro: $L(\lambda)$. Se cumple que: $\{\lambda\}L = L$

- Potencia de un lenguaje:

  Concatenación del mismo lenguaje i veces: $L^i = LLL...L$ (i veces).

  - $L^{i+1} = L^i L \ (i > 0)$
  - $L^i L^j = L^{i+j} \ (i > 0)$
  - $L^0 = \{\lambda\}$

- Clausura positiva de un lenguaje:

  Es el lenguaje obtenido de la unión del lenguaje con todas sus potencias posibles excepto L^0. Si $\lambda \notin L$, la clausura positiva no contiene λ.

  Se representa como: $L^+ = \bigcup_{i=1}^{\infty} L^i$

  Como cualquier alfabeto es un lenguaje sobre sí mismo, al aplicarle esta operación se observa que:

  $$\Sigma^+ = W(\Sigma) - \{\lambda\}$$

- Cierre (o clausura) de un lenguaje: Lenguaje obtenido uniendo el lenguaje con todas sus potencias posibles (incluyendo L^0). Todas las clausuras contienen a la palabra vacía.

  - $L^* = L^+ \cup \{\lambda\}$
  - $L^+ = LL^*$ (imposible obtener la palabra vacía).
  - $W(\Sigma) = L^*$

- Reflexión de un lenguaje:

  Lenguaje que contiene las palabras inversas a las palabras de L.

  $L^{-1} = \{x^{-1} : x \in L\}$

2.2. Gramáticas Formales

2.2.1. Concepto de gramática formal

Define la estructura de las frases y de las palabras de un lenguaje.
Es un método para generar palabras de un lenguaje a partir de un alfabeto mediante **derivaciones**.

Considerando la instrucción $x = y + 2 * z$ con:

- Conjunto de producciones:

  $\ll instruccion \gg ::= \ll asignacion \gg$

  $\ll asignacion \gg ::= \ll identificador \gg$ " = " $\ll expresion \gg$

  $\ll expresion \gg ::= \ll sumando \gg$

  $\ll expresion \gg ::= \ll sumando \gg$ " + " $\ll expresion \gg$

  $\ll sumando \gg ::= \ll factor \gg$

  $\ll sumando \gg ::= \ll factor \gg$ " + " $\ll sumando \gg$

  $\ll factor \gg ::= \ll identificador \gg$

  $\ll factor \gg ::= \ll numero \gg$

- Reglas morfológicas:

  $\ll identificador \gg ::=$ "x"

  $\ll identificador \gg ::=$ "y"

  $\ll identificador \gg ::=$ "z"

  $\ll numero \gg ::=$ "2"

Podemos obtener la expresión anterior concatenando producciones a partir de $\ll instruccion \gg$.

Producción o regla ($x ::= y$): Es un par ordenado con $x, y \in \Sigma^*$. Esto permite transformar palabras en otras.

Derivación Directa ($v \to w$): Aplicación de una producción a una palabra v para convertirla en otra palabra w donde:
$v = zxu,\ w = zyu\ \ (v, w, z, u \in \Sigma^*)$

Derivación $v \to {}^*w$: Aplicación de una secuencia de producciones a una palabra.

Longitud de la derivación: Número de derivaciones que hay que aplicar para obtener una palabra.

Derivación más a la izquierda: Se utiliza en cada derivación directa la producción aplicada a los símbolos más a la izquierda de la palabra.

Derivación más a la derecha: Se utiliza en cada derivación directa la producción aplicada a los símbolos más a la derecha de la palabra.

La **gramática formal** está formada por la cuádrupla: $G = (\Sigma_T, \Sigma_N, S, P)$. Donde:

- Σ_T: Alfabeto de los símbolos terminales.
- Σ_N: Alfabeto de los símbolos no terminales.
- $S \in \Sigma_N$, es el axioma o símbolo inicial.
- P es un conjunto finito de reglas de producción de la forma:
  $u ::= v$, donde $u \in \Sigma^+$ y $v \in \Sigma^*$

Se verifica que:

- $\Sigma_T \cap \Sigma_N = \emptyset$
- $\Sigma = \Sigma_T \cup \Sigma_N$

Si existen dos reglas de la forma:
$u ::= v$
$u ::= w$
Se puede representar de la forma:
$u ::= v|w$ (Forma Normal de Backus (BNF)).

Una palabra $x \in \Sigma^*$ se denomina forma sentencial de una gramática $G = (\Sigma_T, \Sigma_N, S, P)$ si se verifica que: $S \to {}^*x$.

Si se cumple que $x \in \Sigma_T^*$ se dice que x es una sentencia o instrucción de G. Las sentencias están compuestas por símbolos terminales.

Sea una gramática $G = (\Sigma_T, \Sigma_N, S, P)$. Se llama lenguaje generado por G al conjunto:
$L(G) = \{x : S \to {}^*x \wedge x \in \Sigma_T^*\}$ (Conjunto de todas las sentencias de la gramática).

2.2.2. Tipos de gramáticas: Jerarquía de Chomsky

Chomsky clasificó las gramáticas en cuatro grupos. De forma que:
$G3 \subset G2 \subset G1 \subset G0$.

Gramáticas Tipo 0 (Lenguajes sin restricciones)
Tienen la forma: $u ::= v$ donde $u \in \Sigma^+, v \in \Sigma^*, u = xAy$ con $x, y \in \Sigma^*, A \in \Sigma_N$.

- La parte izquierda de las reglas de producción no puede ser λ.
- En la parte izquierda ha de aparecer algún símbolo no terminal.

Como podemos encontrar que la parte derecha sea más corta que la izquierda, puede darse el caso de encontrar alguna regla donde las derivaciones puedan ser decrecientes (**reglas compresoras**).

Gramáticas Tipo 1 (Lenguajes dependientes del contexto)
Las reglas de producción tienen la forma:
$xAy ::= xvy$ donde $x, y \in \Sigma^*, v \in \Sigma^+, A \in \Sigma_N$.

Como v no puede ser λ, este tipo de gramáticas no pueden tener reglas compresoras. Exceptuando la regla $S ::= \lambda$. Como consecuencia,
$\lambda \in L(G) \leftrightarrow \{S ::= \lambda\} \in P$.

Gramáticas Tipo 2 (Lenguajes independientes del contexto)
Las reglas de esta gramática son de la forma:
$A ::= v$ donde $v \in \Sigma^+, A \in \Sigma_N$ y pueden contener la regla $S ::= \lambda$.
La mayor parte de los lenguajes de programación se describen mediante este tipo.
La conversión de A en v no depende del contexto de A.

Gramáticas Tipo 3 (Lenguajes Regulares)
Se pueden clasificar en:

- Gramáticas lineales por la izquierda.

  Las reglas de producción pueden ser de la forma:

  - $A ::= a$
  - $A ::= Va$
  - $S ::= \lambda$

  Donde $a \in \Sigma_T, A, V \in \Sigma_N$ y S es el axioma de la gramática.

- Gramáticas lineales por la derecha.

  Las reglas de producción pueden ser de la forma:

  - $A ::= a$
  - $A ::= aV$
  - $S ::= \lambda$

  Donde $a \in \Sigma_T, A, V \in \Sigma_N$ y S es el axioma de la gramática.

2.2.3. Árboles de Derivación

A toda derivación de una gramática de tipo 1,2 o 3 le corresponde un árbol de derivación que se construye de forma que:

- La **raíz** del árbol sea el axioma de la gramática.
- Una **derivación directa** se representa por un conjunto de ramas que salen de un nodo determinado.
- Los **nodos finales** de cada paso (leídos de izquierda a derecha) forman la forma sentencial obtenida por la derivación.
- Será una **rama terminal** la que esté dirigida hacia un símbolo terminal (hoja). El conjunto de hojas (de izquierda a derecha) forma la sentencia generada por la derivación.

Proceso inverso: Para cada derivación existe un único árbol, pero de una misma sentencia se pueden obtener varias derivaciones diferentes.

2.2.4. Ambigüedad

Una sentencia es ambigua si, para una misma sentencia, podemos obtener varios árboles de derivación diferentes.
Hay lenguajes para los cuales es imposible encontrar gramáticas no ambiguas. (**Lenguajes inherentemente ambiguos**).

2.2.5. Recursividad

Una gramática es recursiva en $A, A \in \Sigma_N$ si:

- $A \rightarrow +xAy$
  - Si $x = \lambda$, entonces se denomina recursiva izquierdas.
  - Si $y = \lambda$, entonces será recursiva a derechas.

  Una gramática recursiva genera un lenguaje infninito.