Lenguajes formales y autómatas

Gramaticas

- Describe la estructura de las frases y de las palabras de un lenguaje mediante reglas.
- Permiten expresar lenguajes infinitos en forma finita
- Las reglas definen ciertos términos en función de otros y se representan mediante la siguiente notación:
    - *{termino que se está definiendo} ::= {definición}*

Reglas de producción:

- <oracion> ::= <sujeto> <predicado>
- <sujeto> ::= <determinante> <sustantivo>
- <predicado> ::= <verbo> <complemento>
- <complemento> ::= <determinante> <sustantivo>

Reglas morfológicas:

- <sustantivo> ::= "hombre"
- <sustantivo> ::= "libro"
- <determinante> ::= "el"
- <determinante> ::= "un"
- <verbo> ::= "lee"

- **Derivación directa** $u \rightarrow v$
  - Es la aplicación de una regla para obtener una palabra a partir de otra
  - Se dice que *v* deriva directamente de *u*, si **u = xyz**, aplicando la regla **y ::= w** se llega a **v = xwz**
- **Derivación** $u \rightarrow^{+} v$
  - Es la aplicación de más de una regla para obtener una palabra a partir de otra
  - Se dice que *v* deriva de *u*, si $u = u_0 \rightarrow u_1 \rightarrow u_2 \rightarrow u_n = v$
- **Relación de Thue** $u \rightarrow^{*} v$
  - Existe una relación de Thue entre u y v si **u = v** o $u \rightarrow^{+} v$

Definición formal de gramática

- Se llama gramática formal a la cuádrupla: $G=(\Sigma_T,\Sigma_N,S,P)$
- Donde:
  - Σ_T es el alfabeto de símbolos de **Terminales**
  - Σ_N es el alfabeto de símbolos de **No Terminales**
  - S es el elemento **distinguido** o **axioma**
  - P es un conjunto finito de producciones

- **Notación de Backus (BNF)**
  - notación abreviada para reglas que comparten la parte izquierda

u ::= v u ::= w	u ::= v **|** w

- **Forma sentencial**
  - Sea una palabra $x \in (\Sigma_T \bigcup \Sigma_N)^*$, donde $S \to^* x$; x es una forma sentencial
- **Sentencia**
  - Sea una palabra $x \in \Sigma_T^*$, donde $S \to^* x$; x es una sentencia

- **Lenguaje asociado a una gramática** o **lenguaje generado por una gramática**
    - Se denomina así al conjunto de todas las sentencias de G
    - $L(G) = \{x | x \in \Sigma_T^* \wedge S \rightarrow^* x\}$
    - Dos gramáticas son **equivalentes** cuando describen el mismo lenguaje
- **Recursividad**
    - una producción es recursiva si posee la forma U ::= x U y
    - Si x = λ la gramatica es **recursiva a izquierda**
    - Si y = λ la gramatica es **recursiva a derecha**
    - Si un lenguaje es infinito, la gramática que lo representa tiene que ser recursiva.

Clasificación de Chomsky

- Tipo 0: Gramática sin restricciones
  - $u ::= v$
  - $u = xAy,\ x, y, v \in (\Sigma_T \bigcup \Sigma_N)^*\ y\ A\ \in \Sigma_N$
- Tipo 1: Gramática sensible al contexto
  - $xAy ::= xvy,\ x, y \in (\Sigma_T \bigcup \Sigma_N)^*, v \in (\Sigma_T \bigcup \Sigma_N)^+\ y\ A\ \in \Sigma_N$
  - No se admiten derivaciones en λ
- Tipo 2: Gramática independiente al contexto
  - $A ::= v,\ v \in (\Sigma_T \bigcup \Sigma_N)^*\ y\ A\ \in \Sigma_N$
  - No se admiten derivaciones en λ

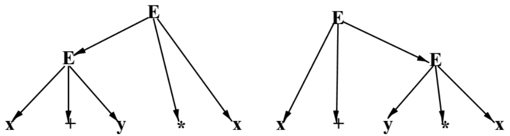
- Tipo 3: Gramática regular o linear
  - Aceptan 3 tipos de producciones
  - Lineales por la izquierda
    - $A::=a$
    - $A::=Va$
    - $S::=\lambda$
  - Lineales por la derecha
    - $A::=a$
    - $A::=aV$
    - $S::=\lambda$

Arbol de derivación

Representación gráfica de las derivaciones para gramáticas de tipo 1, 2 o 3

- La raíz del árbol se etiqueta con el axioma de la gramática.
- Por cada derivación directa, desde el nodo etiquetado con el símbolo terminal que se sustituye se hace surgir un conjunto de arcos que se dirigen a nodos etiquetados con los símbolos de la cadena por que se sustituye.
- Se denomina **subárbol** a una parte del árbol de derivación que pende de un nodo asociado a un no terminal que incluye todos los nodos que descienden del mismo

1. $E \rightarrow E + E$
2. $E \rightarrow E * E$
3. $E \rightarrow x$
4. $E \rightarrow y$



Gramáticas ambiguas

- Una gramática es ambigua si posee **al menos** una sentencia ambigua
- Una sentencia es ambigua cuando es posible obtenerla mediante más de un árbol de derivación. (Ver fig. anterior)
- Un lenguaje es **inherentemente ambiguo** si no es posible representarlo mediante una gramática no ambigua

Gramáticas limpias y bien formadas

1. Regla de producción **innecesaria**

   - Es de la forma $U ::= U$
   - Hacen la gramática ambigua y no aportan a la generación de palabras
   - Estas reglas deben eliminarse

2. Símbolo **inaccesible desde el axioma**

- No es el axioma y no aparece en la parte derecha de ninguna de las reglas alcanzables desde el axioma
- Todo símbolo σ accesible desde el axioma cumple que $S \rightarrow^* x\sigma y; x, y \in \Sigma^*$

2.1. Hacer una lista de los símbolos de la gramática (*T* y *NT*) y marcar el distinguido

2.2. Por cada regla de la forma $U ::= u$, donde *U* está marcado, marcar todos los símbolos de la derecha

2.3. Repetir 2.2 hasta que no se marque ningún símbolo

2.4. Eliminar todos los símbolos no marcados de los alfabetos

2.5. Eliminar todas las producciones que contengan alguno de estos símbolos

3. No terminal **no generativo**

- Cuando el lenguaje generado a partir de ese símbolo es el vacío
- un simbolo **U** no es **no generativo** si $U \to^{+} u; u \in \Sigma_T^*$
- toda regla que contenga un símbolo no generativo se denomina **regla superflua**

3.1. Hacer una lista de los símbolos no terminales de la gramática

3.2. Por cada regla de la forma $U ::= u$, donde *u* está formada únicamente por terminales y no terminales marcados, marcar **U**

3.3. Repetir 2.2 hasta que no se marque ningún símbolo

3.4. Eliminar todos los símbolos no marcados del conjunto de no terminales

3.5. Eliminar todas las producciones que contengan alguno de estos símbolos

4. Gramática **reducida** es aquella que no posee símbolos inaccesibles desde el axioma, símbolos no generativos ni reglas superfluas.

5. Una gramática está **limpia** si es **reducida** y no posee reglas innecesarias.

Ejemplo (el desarrollo está en el libro)

Gramática original	Gramática limpia
S ::= PQ | aSb | S | P | R P ::= aPQ | a Q ::= Qb | λ R ::= Rb U ::= aP | b	S ::= PQ | aSb | P P ::= aPQ | a Q ::= Qb | λ

6. Las reglas de la forma $U::=\lambda$ son **reglas no generativas**

- Si el lenguaje no posee la palabra vacía, se pueden eliminar todas, sino solo hay que dejar la palabra vacía en el axioma

6.1. Tomar una regla de la forma $U::=\lambda$ y eliminarla de la gramática

6.2. Por cada regla de la gramática donde *U* aparece en la parte derecha, $V::=xUy$, añadir la regla $V::=xy$ (a menos que esta exista)

6.3. Repetir hasta que no haya reglas que deriven en lambda o solo quede una, siendo el axioma la parte izquierda de la misma

Ejemplo anterior	Sin reglas no generativas
S ::= PQ | aSb | P P ::= aPQ | a Q ::= Qb | λ	S ::= PQ | aSb | P P ::= aPQ | aP | a Q ::= Qb | b

7. Las reglas de la forma $U::=V$ son **reglas de redenominación**
   7.1. Tomar una regla de la forma $U::=V$ y eliminarla de la gramática

   7.2. Por cada regla de la gramática de la forma $V::=x$, añadir la regla $U::=x$ (a menos que esta exista)

   7.3. Repetir hasta que no haya reglas de redenominación

   - Este algoritmo puede introducir reglas innecesarias.

Ejemplo anterior	Gramática limpia
S ::= PQ | aSb | P P ::= aPQ | aP | a Q ::= Qb | b	S ::= PQ | aSb | aPQ | aP | a P ::= aPQ | aP | a Q ::= Qb | b

8. Gramática **bien formada** es aquella que está limpia y no posee reglas no generativas o de redenominación.

Bibliografía y enlaces útiles.

- Alfonseca Cubero y otros - Teoría de autómatas y lenguajes formales - McGRAW-HILL