**Jerarquía de Chomsky**

Es una clasificación jerárquica de distintos tipos de gramáticas formales que generan lenguajes formales.

La Jerarquía de Chomsky consta de cuatro niveles:

* **Tipo 0:** **Lenguajes recursivos:**

Conjuntos de objetos formales de cualquier complejidad computacional.

*Ejemplo:*

a. Daniela me ha dicho que Itzel vendrá.

b. Daniela me ha dicho que Emanuel piensa que Itzel vendrá.

c. Daniela me ha dicho que Emanuel piensa que Pepe considera que Itzel vendrá.

* **Tipo 1: Lenguajes sensibles al contexto:**

Conjuntos de conjuntos de secuencias de símbolos (o “cadenas”).

*Ejemplo:*

Ejemplo 5

L5={a^i b^j c^i d^j / i, j ≥0 } G5=<{A, B, C}, {a, b, c}, S5, P5> donde P5 contiene las siguientes producciones:

G5=<{A, B, C}, {a, b, c}, S5, P5> donde P5 contiene las siguientes: producciones:

S5→ ε DC→CD

S5→ A bC→bc

A→aAC cC→cc

A→ac cD→cd

A→B dD→dd

B→bBD bD→bd

B→ bD

* **Tipo 2: Lenguajes libres de contexto:**

Conjuntos de secuencias de símbolos (o “frases”).

*Ejemplo:*

• Ejemplo: Gpal = ({P}, {0, 1}, A, P), donde

A = {P → , P → 0, P → 1, P → 0P0, P → 1P1}.

Muchas veces se agrupan las producciones con la misma

cabeza, e.g., A = {P → |0|1|0P0|1P1}.

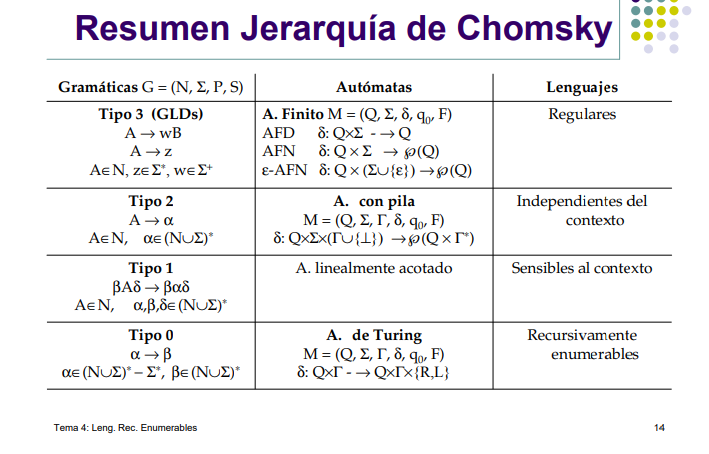
* **Tipo 3: Lenguajes regulares:**

Secuencias de símbolos.

*Ejemplo:*

Dado Σ = {a,b}, las siguientes afirmaciones son ciertas:

* φ y {ε} son lenguajes regulares
* {a} y {b} son lenguajes regulares
* {a, b} es un lenguaje regular
* {ab} es un lenguaje regular
* {a, ab, b} es un lenguaje regular
* {a^i | i ≥ 0} es un lenguaje regular
* {a^i b^j | i ≥ 0 y j ≥ 0} es un lenguaje regular
* {(ab)^i | i ≥ 0} es un lenguaje regular



**YACC**

Yacc es un programa para generar analizadores sintácticos. Las siglas del nombre significan Yet Another Compiler-Compiler, es decir, "Otro generador de compiladores más". Genera un analizador sintáctico basado en una gramática analítica escrita en una notación similar a la BNF.

* Stephen C. Johnson. YACC: Yet another compiler-compiler. *Unix Programmer's Manual* Vol 2b, 1979.

**LEX (o FLEX)**

* Flex es una herramienta para construir analizadores léxicos.
* Flex recibe como entrada un conjunto de descripciones de tokens, y genera la función C yylex() que es un analizador léxico que reconoce dichos tokens.
* Los tokens se describen mediante patrones que son extensiones de las expresiones regulares.
* Al conjunto de las descripciones de tokens se le llama especificación Flex.

# Referencias

Aho, A. V., S. Lam, M., Sethi, R., & D. Ullman, J. (2008). *Compiladores.* Ciudad de México: PEARSON EDUCACIÓN.

Álvarez Pérez-Aradros, P. J., & Béjar Hernández, R. (2003). *Lenguajes Regulares.* Obtenido de Departamento de Informática e Ingeniería de Sistemas: http://webdiis.unizar.es/asignaturas/LGA/material\_2010\_2011/3\_lenguajesRegulares.pdf

Aranda, J., Duro, N., Fernández, J. L., Jiménez, J., & Morilla, F. (2006). *Fundamentos de Lógica Matemática y Computación.* Madrid: Sanz y Torres.

De la Cruz, M., & Ortega, A. (2007). *Construcción de un analizador léxico para ALFA con Flex.* Obtenido de Prácticas de Procesadores de Lenguaje 2007-2008: http://arantxa.ii.uam.es/~mdlcruz/docencia/compiladores/2007\_2008/lexico\_07\_08.pdf

Ibarra Florencio, N. (Agosto de 2011). *Tipos de Lenguajes Formales.* Obtenido de Centro de Ciencias de la Complejidad: http://www.comunidad.escom.ipn.mx/genaro/Papers/Veranos\_McIntosh\_files/lenguajesNivardo.pdf

Navarro, M. (3 de Noviembre de 2010). *Tema 4: Lenguajes recursivos y.* Obtenido de Modelos Abstractos de Cómputo I (MAC I) y Autómatas y Lenguajes Formales (ALF): http://www.sc.ehu.es/jiwnagom/MAC1-ALF/MAC-archivos/Tema4.pdf