Universidad Politécnica de Aguascalientes

Materia: Lenguajes y Autómatas.

Tarea: Ensayo Unidad 4.

Profesor: Dr. Christian José Correa Villalón.

Alumno: Juan Carlos Pedroza Hernández.

Matrícula: UP170132.

Grupo: ISEI007.

Fecha de Entrega: 17/04/2020

Contenido

• Gramáticas y la jerarquía de CHOMSKY

En 1956 Chomsky crea una manera para clasificar los lenguajes formales en cuatro categorías enumerados del 0 al 3 y los mecanismos organizados formalizadores como gramáticas formales y expresiones y autómatas para reconocer cada tipo.

También se conoce bajo el nombre de *Clasificación de Chomsky* o *Jerarquía matemática de los lenguajes*.

Los autómatas son mecanismos formales que "realizan" derivaciones en gramáticas formales. La manera en que las realizan es mediamente el reconocimiento. Una palabra se genera en una gramática si y solo sí cumple con las condiciones terminales del autónomo. Por esto es que los autómatas son analizados los léxicos de las gramáticas que les corresponde

Se puede caracterizar qué tan compleja es una gramática formal. Noam Chomsky mostró que esta caracterización clasifica jerárquicamente a las gramáticas formales: gramáticas en un nivel están incluidas en los siguientes niveles y la inclusión entre niveles es propia.

La clasificación de lenguajes en clases de lenguajes es debida a N. Chomsky, quien propuso una jerarquía de lenguajes, donde las clases más complejas incluyen a las más simples.

- Los "Lenguajes Regulares", es la clase más pequeña, e incluye a los lenguajes más simples. Un ejemplo de lenguaje regular es el conjunto de todos los números binarios.
- Los "Lenguajes Libres de Contexto", que incluyen a los Lenguajes Regulares. Por ejemplo, la mayoría de los lenguajes de programación son Lenguajes Libres de Contexto.
- Los "Lenguajes Recursivamente y numerables", que incluyen a los Libres de Contexto (y por lo tanto a los Lenguajes Regulares).

La Jerarquía de Chomsky tiene cuatro niveles:

Gramáticas 0 (sin restricciones): que incluye a todas las gramáticas formales. Estas gramáticas generan todos los lenguajes capaces de ser reconocidos por una máquina de Turing. Los lenguajes son conocidos como lenguajes recursivamente numerables.

Las gramáticas que generan estos lenguajes pueden tener reglas compresoras. Las reglas de producción son de la siguiente forma:

$$\mathrm{P} = \{(u \to v) | u = xAy; u \in \Sigma^+; v, x, y \in \Sigma^*; A \in N\}$$

Ilustración 1: Tipo 0.

Gramáticas 1 (gramáticas sensibles al contexto): generan los lenguajes sensibles al contexto. Estas gramáticas tienen reglas de la forma con A un no terminal y “α, β y” γ cadenas de terminales y no terminales. Las cadenas α y β pueden ser vacías, pero “γ” no puede serlo.

La regla está permitida si S no aparece en la parte derecha de ninguna regla. Los lenguajes descritos por estas gramáticas son exactamente todos aquellos lenguajes reconocidos por una máquina de Turing determinista cuya cinta de memoria está acotada por un cierto número entero de veces sobre la longitud de entrada, también conocidas como autómatas linealmente acotados.

No existen reglas compresoras en toda la teoría, salvo, opcionalmente, la que deriva el axioma a la palabra vacía.

Existen reglas en las que un símbolo no terminal puede derivar a formas sentenciales distintas, según los símbolos que aparezcan a su alrededor.

Las reglas de producción son de la siguiente forma:

$$\mathrm{P} = \{(S \to \lambda) o (xAy \to xvy) | v \in \Sigma^+; x, y \in \Sigma^*; A \in N\}$$

Ilustración 2: Tipo 1.

Gramáticas 2 (gramáticas libres del contexto): generan los lenguajes independientes del contexto. Las reglas son de la forma con A un no terminal y γ una cadena de terminales y no terminales. Estos lenguajes son aquellos que pueden ser reconocidos por un autómata con pila.

La mayoría de los lenguajes de programación entran en ésta categoría en cuanto su forma sintáctica, aunque en realidad los lenguajes de programación son dependientes del contexto, se reconocen a través de lenguajes de tipo 2 porque su reconocimiento es de O(n) mientras que los de tipo 1 tienen un orden de reconocimiento O(n^3) en el peor caso. Por este motivo se ejecuta un análisis semántico para reconocer si el programa es correcto.

Las reglas de producción son de la siguiente manera:

$$\mathrm{P} = \{(S \to \lambda) o (A \to v) | v \in \Sigma^+; A \in N\}$$

Ilustración 3: Tipo 2.

Gramáticas 3 (gramáticas regulares): generan los lenguajes regulares. Estas gramáticas se restringen a aquellas reglas que tienen en la parte izquierda un no terminal, y en la parte derecha un solo terminal, posiblemente seguido de un no terminal. La regla también está permitida si S no aparece en la parte derecha de ninguna regla. Estos lenguajes son aquellos que pueden ser aceptados por un autómata finito. También estas familias de lenguajes pueden ser obtenidas por medio de expresiones regulares.

Son los lenguajes más simples dentro de la Jerarquía de Chomsky. Se suelen expresar mediante expresiones regulares.

Existen 2 tipos: lineales por la derecha y lineales por la izquierda. Las reglas de producción son de la siguiente forma:

Lineales por la derecha:

$$\mathrm{P} = \{(S \to \lambda) o (A \to aB) o (A \to a) | a \in T; A, B \in N\}$$

Lineales por la izquierda:

$$\mathrm{P} = \{(S \to \lambda) o (A \to Ba) o (A \to a) | a \in T; A, B \in N\}$$

Ilustración 4: Tipo 3.

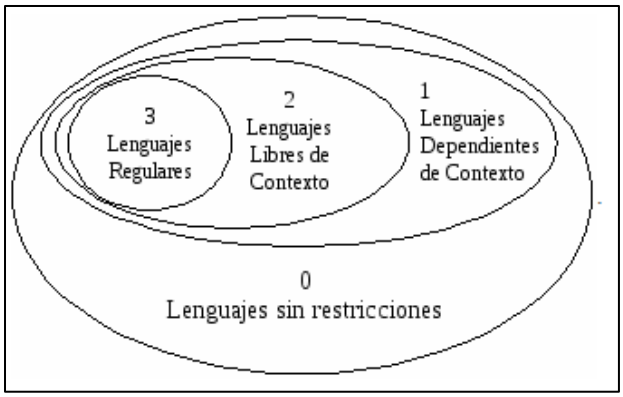

Ilustración 5: Lenguajes.

La jerarquía de Chomsky no solo aporta un ordenamiento conjunto de los lenguajes, sino que proporciona un mecanismo de clasificación basado en características relacionadas con las formas gramaticales mínimas de cada lenguaje en particular, así como decide qué tipo de reconocedores mínimos servirían para determinar las cadenas válidas.

Tabla 1: Gramáticas.

TIPO	NOMBRE	FORMA DE LAS PRODUCCIONES	TIPO DE AUTÓNOMAS O RECONOCEDORES
0	LRE	IRRESTRICTAS	Máquina de Turing
1	LDC	$xAy \rightarrow xzy$	Autómatas Linealmente acotados
2	LLC	$A \rightarrow x$	Autómata de pila
3	LR	$A \rightarrow x$	Autómata finito Expresión regular

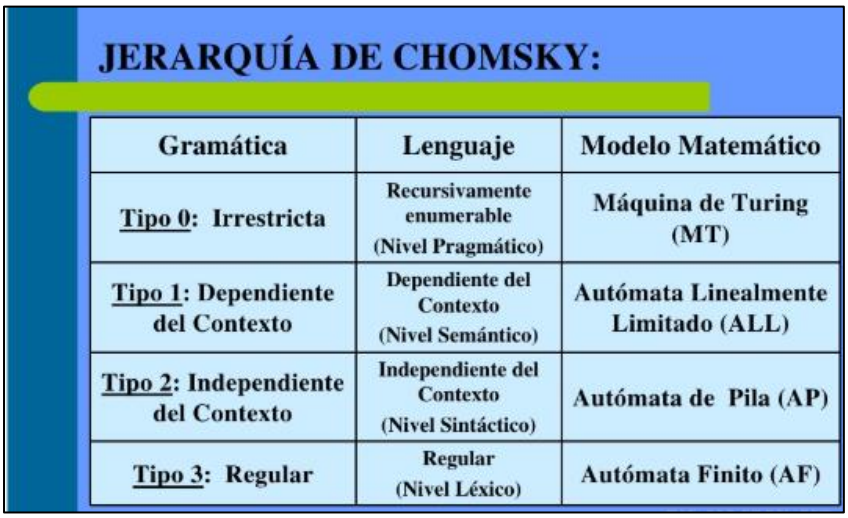

Ilustración 6: Jerarquía de Chomsky.

- ## Arboles de Derivación

Un árbol de derivación (o *árbol sintáctico*) es una representación gráfica de cómo se deriva una forma sentencial a partir del símbolo no-terminal inicial.

Un árbol es un grafo dirigido acíclico en el cual cada nodo se conecta con un nodo distinguido, llamado nodo raíz por un único camino. Un nodo n_1 se dice descendiente de otro nodo n_2 si se puede llegar a n_1 a partir de n_1. El nodo raíz no es descendiente de ningún nodo, y los nodos que no tienen descendientes se denominan hojas. El resto de los nodos se denominan nodos interiores.

Un árbol de derivación permite mostrar gráficamente cómo se puede derivar cualquier cadena de un lenguaje a partir del símbolo distinguido de una gramática que genera ese lenguaje.

Un árbol es un conjunto de puntos, llamados nodos, unidos por líneas, llamadas arcos. Un arco conecta dos nodos distintos. Para ser un árbol un conjunto de nodos y arcos debe satisfacer ciertas propiedades:

- Hay un único nodo distinguido, llamado raíz (se dibuja en la parte superior) que no tiene arcos incidentes.
- Todo nodo c excepto el nodo raíz está conectado con un arco a otro nodo k, llamado el padre de c (c es el hijo de k). El padre de un nodo, se dibuja por encima del nodo.
- Todos los nodos están conectados al nodo raíz mediante un único camino.
- Los nodos que no tienen hijos se denominan hojas, el resto de los nodos se denominan nodos interiores.

El árbol de derivación tiene las siguientes propiedades:

- El nodo raíz está rotulado con el símbolo distinguido de la gramática;
- Cada hoja corresponde a un símbolo terminal o un símbolo no terminal;
- Cada nodo interior corresponde a un símbolo no terminal.

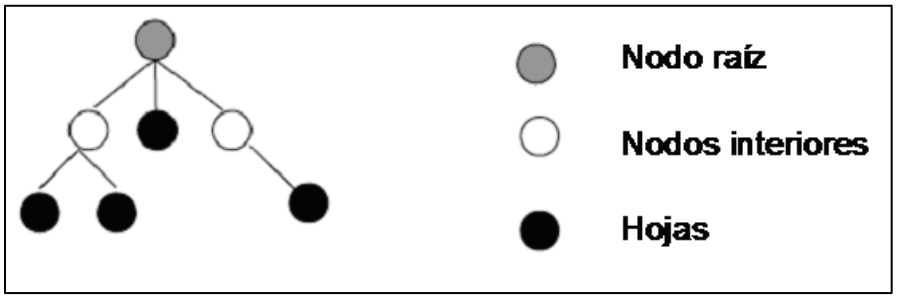

Ilustración 7: Árbol de derivación.

Construcción de un árbol de derivación:

- Cada nodo del árbol va a contener un símbolo.
- En el nodo raíz se pone el símbolo inicial S.
- Se efectúa una ramificación del árbol por cada producción que se aplique: Si a la variable de un nodo A se le aplica una determinada regla A ->α, entonces para cada símbolo que aparezca en α se añade un hijo con el símbolo correspondiente, situados en el orden de izquierda a derecha.
- Este proceso se repite para todo paso de la derivación. Si la parte derecha es una cadena vacía, entonces se añade un solo hijo, etiquetado con ε. En cada momento, leyendo los nodos de izquierda a derecha se lee la palabra generada.

Ejemplo 1:

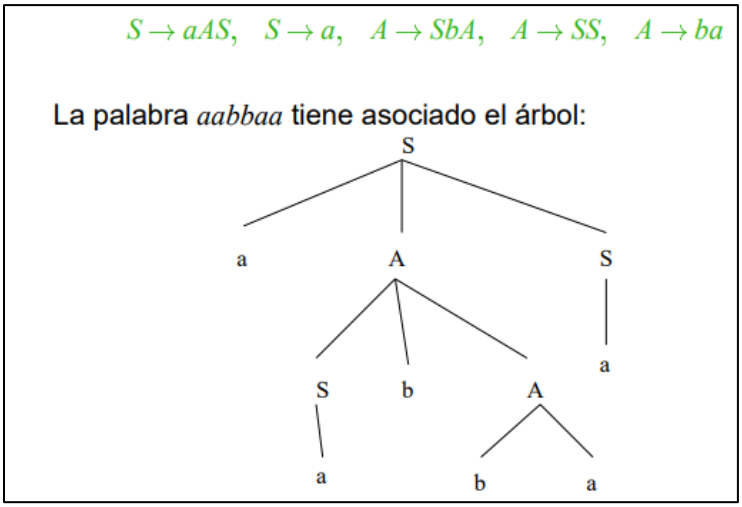

Ilustración 8: Ejemplo de creación de árbol de derivación

Un árbol de derivación puede proceder de dos cadenas de derivación distintas:

- Se llama derivación por la izquierda asociada a un árbol a aquella en la que siempre se deriva primero la primera variable (más a la izquierda) que aparece en la palabra.
- Se llama derivación por la derecha asociada a un árbol a aquella en la que siempre se deriva primero la última variable (más a la derecha) que aparece en la palabra.

Ejemplo 2:

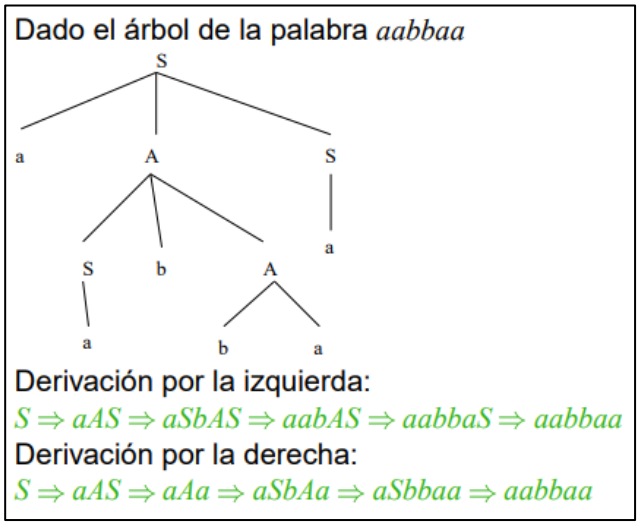

Ilustración 9: Tipos de derivación.

Gramática ambigua

Una gramática se dice ambigua si existe una palabra con dos árboles de derivación distintos.

Ejemplo:

La gramática S ->AA, A-> aSa, A ->a es ambigua, ya que la palabra a^5 tiene los dos árboles siguientes.

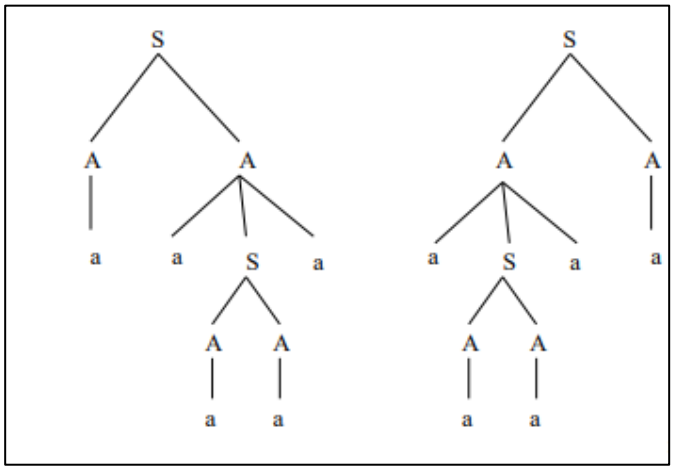

Ilustración 10: Gramática ambigua.

Lenguaje inherentemente ambiguo:

Un lenguaje de tipo 2 es inherentemente ambiguo si toda gramática que lo genera es ambigua.

Ejemplo:

El lenguaje generado por la gramática anterior S->AA, A->aSa, A-> a, no es inherentemente ambiguo. Este lenguaje es $\{a^{2+3i}: i \geq 0\}$ y puede ser generado por la gramática:

S->aa, S ->aaU, U ->aaaU, U ->aaa, que no es ambigua. Único árbol para a^5:

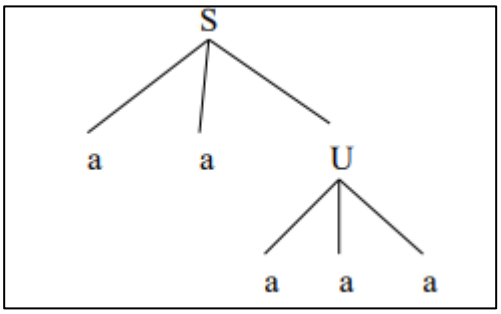

Ilustración 11: Lenguaje inherentemente ambiguo

• Pruebas de corrección

Es posible en general hacer pruebas matemáticas de que una gramática corresponde a un lenguaje dado. Esto tiene la gran ventaja de que dicha correspondencia ya no es una simple convicción intuitiva, sino que adquiere el rango de certeza matemática. En ciertas aplicaciones, donde es extremadamente importante asegurarse de que no hay errores, las pruebas que garantizan esta certeza son de un gran valor.

Las pruebas que permiten establecer la correspondencia entre un lenguaje y una gramática dados requieren dos partes:

1. Prueba de corrección, que garantiza que todas las palabras que se producen al utilizar la gramática efectivamente corresponden a la descripción del lenguaje dado;
2. Prueba de completez, que se asegura de que, al producir palabras con la gramática, no falten palabras del lenguaje dado.

En general las pruebas de corrección son más sencillas y siguen un patrón más predecible que las de completez. Las pruebas de corrección se hacen por inducción, más precisamente por inducción en la longitud de la derivación.

La idea de una prueba por inducción basada en la longitud de la derivación es esencialmente mostrar que todas las palabras por las que se pasa en medio del proceso de derivación cumplen una propiedad, que es básicamente el enunciado del lenguaje. Dichas pruebas siguen el siguiente esquema:

1. Lo primero que hay que hacer es establecer un enunciado, relacionado con la dentición del lenguaje considerado, pero algo medicado de manera que se pueda aplicar a las palabras intermedias en el proceso de derivación, las cuales pueden contener variables tanto como constantes.
2. Luego se prueba, como base de la inducción, que para las palabras intermedias de la derivación producidas en al menos k0 pasos, la propiedad se cumple.
3. A continuación, se hace el paso de inducción propiamente dicho. Para esto primero se supone que la propiedad se cumple tras haber hecho i pasos de derivación (esto es la hipótesis de inducción), y luego se prueba que también se cumple al hacer un paso más de derivación (esto es, para las palabras derivadas en i+1 pasos). Al concluir este paso, se ha probado que todas las palabras intermedias en el proceso de derivación cumplen con la propiedad.
4. Finalmente, hay que particularizar la propiedad para la última palabra de la derivación, que es la que sólo contiene constantes. Con esto se termina la prueba.

Bibliografía

http://delta.cs.cinvestav.mx/~gmorales/ta/node11.html. (s.f.). Obtenido de http://delta.cs.cinvestav.mx/~gmorales/ta/node11.html.

http://lenguajesyautomatas1unidad6-2.blogspot.com/2016/06/arboles-de-derivacion.html. (s.f.). Obtenido de http://lenguajesyautomatas1unidad6-2.blogspot.com/2016/06/arboles-de-derivacion.html.

http://www.suigeneris.org/UCABTI/Arboles%20de%20Derivacion.html. (s.f.). Obtenido de http://www.suigeneris.org/UCABTI/Arboles%20de%20Derivacion.html.

https://es.slideshare.net/LuisCouoh/arboles-de-derivacion. (s.f.). Obtenido de https://es.slideshare.net/LuisCouoh/arboles-de-derivacion.

https://ocw.unican.es/pluginfile.php/1516/course/section/1946/1-4_Jerarquia_Chomsky.pdf. (s.f.). Obtenido de https://ocw.unican.es/pluginfile.php/1516/course/section/1946/1-4_Jerarquia_Chomsky.pdf.

https://www.ecured.cu/Jerarqu%C3%ADa_de_Chomsky. (s.f.). Obtenido de https://www.ecured.cu/Jerarqu%C3%ADa_de_Chomsky.