TECNOLÓGICO NACIONAL DE MEXICO

INSTITUTO TECNOLÓGICO DE IZTAPALAPA

INTEGRANTES:

CUANENEMI CUANALO MARIO ALBERTO	181080030
FERMIN CRUZ ERIK	181080007
GUTIERREZ ARELLANO RAFAEL	181080022
PEREZ ARMAS FAUSTO ISAAC	181080037

ISC-6AM

LENGUAJES Y AUTOMATAS I

M.C. ABIEL TOMÁS PARRA HERNÁNDEZ

SEP 2020 / FEB 2021

ACTIVIDAD SEMANA 5

Cuanenemi Cuanalo Mario Alberto

Representación finita del lenguaje

Bueno ya una vez expuesto el tema en el video y checado información extra para la comprensión del tema nos lleva a Representación finita del lenguaje
Un lenguaje consiste de un grupo de cadenas de un alfabeto.Usualmente ciertas restricciones se aplican a las cadenas del lenguaje. Por ejemplo el lenguaje Español consiste de todas las cadenas de palabras que nosotros llamamos oraciones.No todas las combinaciones de palabras forman oraciones. De ahí que un lenguaje consiste de un subconjunto del conjunto de todas las posibles cadenas que se pueden formar del alfabeto.

Ejemplo: El lenguaje L de cadenas de el alfabeto {a,b} en

donde cada cadena comienza con una a y tiene longitud par.

Las cadenas aa, ab, aaaa, abbb, abab, abbbaaba forman parte

de ese lenguaje.

Palíndromos

Cadenas que se leen igual de izquierda a derecha y viceversa.

Un alfabeto

Es un conjunto finito de símbolos.

Un Lenguaje:

Conjunto de cadenas de símbolos tomados de algún alfabeto.

Lectura:

Sobre el alfabeto {0,1} es seria la siguiente {1,0}.

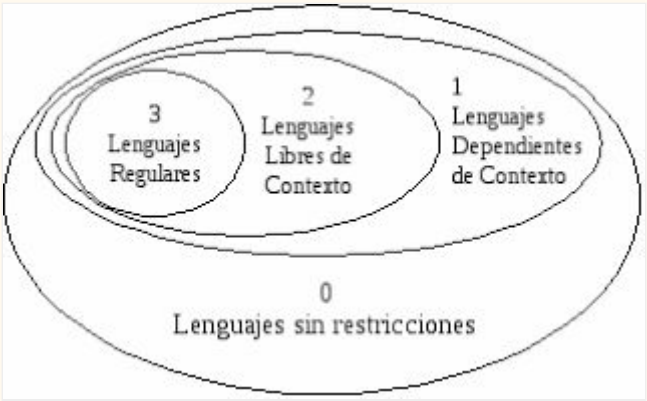
Expresiones Regulares

Los lenguajes aceptados por un autómata finito se describen con facilidad mediante expresiones simples llamadas expresiones regulares.

Sea S un alfabeto. La expresión regular sobre S y los conjuntos que denotan se definen de manera recursiva.

Æ es una expresión regular y denota al conjunto vacío. Es una expresión regular y denota al conjunto {e }. Para cada a Î S , a es una expresión regular y denota al conjunto {a}

TIPOS DE GRAMÁTICA



Ø **TIPO "0" O "No restringida o recursivamente enumerables":**

Incluye a todas las gramáticas formales. Estas gramáticas generan todos los lenguajes de ser reconocidos por una maquina de Turing. Y con este lenguaje se hacen los parser de un compilador

CARACTERÍSTICAS: De este tipo es que del lado derecho de cada producción puede empezar con un símbolo terminal o con un no terminal y del lado izquierdo puedes empezar con más de un símbolo no terminal.

RESTRICCIONES: Es que no tiene solamente que el del lado izquierdo debe haber por lo menos un símbolo no terminal

NOTA: "+" significa "*sin incluir la cadena vacía"* y significa "*incluyendo la cadena vacía". "l"* significa *"o".*

Estos lenguajes también son denominados “recursivamente enumerables”

$$x \rightarrow y$$
$$x \in (NT / T)^{+}$$
$$y \in (NT / T)^{*}$$

Ø TIPO “1” O “Sensible al contexto”:

Estos tipos de lenguajes se resuelven mediante autómatas lineales limitados. Con este tipo se hacen los parser (analizador sintáctico) de un compilador que transforma su entrada en un árbol de derivación.
El analizador sintáctico convierte el texto de entrada en otras estructuras (comúnmente árboles), que son más útiles para el posterior análisis y capturan la jerarquía implícita de la entrada

CARACTERÍSTICAS: Del lado derecho de cada producción puede empezar con un símbolo terminal o con un no terminal y del lado izquierdo puede empezar con más de un símbolo no terminal.

RESTRICCIONES: el número de no terminales del lado izquierdo de la producción debe ser menor o igual al número de símbolos del lado derecho

NOTA: Los lenguajes regulares y los libres de contexto también se puede resolver mediante autómatas lineales limitados

$$\alpha \rightarrow \beta \,;\, |\alpha| \leq |\beta|$$
$$\alpha = z_1 x\, z_2$$
$$\beta = z_1 y\, z_2$$
$$z_1,\, z_2 \in T^{*}$$
$$x \in NT$$
$$y \in (NT / T)^{+}$$

Ø TIPO “2” O “Libres o Independientes de contexto”:

Estos tipos de lenguajes se resuelven mediante autómatas descendentes y con este tipo de lenguaje se programa los parser en un compilador, permiten describir la mayoría de los lenguajes de programación, de hecho, la sintaxis de la mayoría de los lenguajes de programación está definida mediante gramáticas libres de contexto.

CARACTERISTICAS: Del lado derecho de cada producción puede empezar con símbolo terminal o con un no terminal

Los lenguajes regulares también se pueden resolver mediante autómatas descendentes

$$x \rightarrow y$$
$$x \in NT$$
$$y \in (NT/T)^*$$

Ø TIPO "3" O "Lenguajes regulares":

Estos tipos de lenguajes se resuelven mediante autómatas finitos y con este tipo de lenguaje se hacen los scanners. Estas gramaticas se restrigen 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 y también esta familia de lenguajes pueden ser obtenidas por medio de expresiones regulares

CARACTERÍSTICAS: Del lado derecho de cada producción debe empezar con un símbolo terminal.

$$\alpha \rightarrow \beta$$
$$\alpha \in NT$$
$$\beta \in \begin{cases} aB \\ Ba \\ b \end{cases}$$
$$B \in NT$$
$$a \in T^+$$
$$b \in T^*$$

Árboles de Derivación

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.

Esquema de un árbol de derivación

Propiedades de un árbol de derivación:

Sea G = (N,T,S,P) una gramática libre de contexto, sea una variable. Diremos que un árbol TA = (N,E) etiquetado es un árbol de derivación asociado a G si verifica las propiedades siguientes:

- La raíz del árbol es un símbolo no terminal.
- Cada hoja corresponde a un símbolo terminal o λ.
- Cada nodo interior corresponde a un símbolo no terminal.

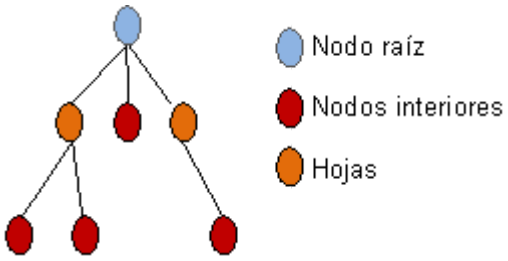
Para cada cadena del lenguaje generado por una gramática es posible construir (al menos) un árbol de derivación, en el cual cada hoja tiene como rótulo uno de los símbolos de la cadena.

Propiedades de un árbol de derivación.

Sea G = (N,T,S,P) una gramática libre de contexto, seauna variable. Diremos que un árbol TA = (N,E) etiquetado es un árbol de derivación asociado a G si verifica las propiedades siguientes:

· La raíz del árbol es un símbolo no terminal
· cada hoja corresponde a un símbolo terminal o λ.
· cada nodo interior corresponde a un símbolo no terminal.

Para cada cadena del lenguaje generado por una gramática es posible construir (al menos) un árbol de derivación, en el cual cada hoja tiene como rótulo uno de los símbolos de la cadena.



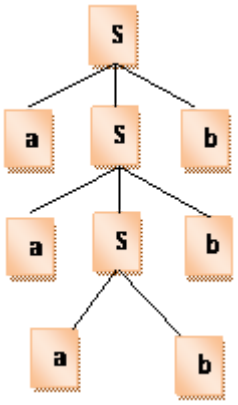
Para cada cadena del lenguaje generado por una gramática es posible construir (al menos) un árbol de derivación, en el cual cada hoja tiene como rótulo uno de los símbolos de la cadena.

Si un nodo está etiquetado con una variable X y sus descendientes (leídos de izquierda a derecha) en el árbol son X1,…,Xk , entonces hay una producción X → X1…Xk en G.

Sea G=(N,T,S,P) una GLC. Un árbol es un árbol de derivación para G si:
1. Todo vértice tiene una etiqueta tomada de
2. La etiqueta de la raíz es el símbolo inicial S
3. Los vértices interiores tienen etiquetas de N
4. Si un nodo n tiene etiqueta A y n1n2...nk respectivamente son hijos del vértice n, ordenados de izquierda a derecha, con etiquetas x1,x2..xk respectivamente, entonces: A→ x1x2...xk debe ser una producción en P
5. Si el vértice n tiene etiqueta λ, entonces n es una hoja y es el único hijo de su padre.

Árbol de derivación. Ejemplo
Sea G=(N,T,S,P) una GLC con P: S→ ab|aSb

La derivación de la cadena aaabbb será:
y el árbol de derivación:

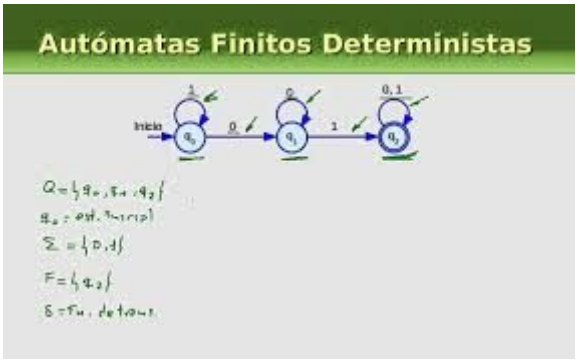


Fermin Cruz Erik

bueno en estos videos siguientes nos dan la continuidad de los que ya habíamos visto y que nos hablaban sobre lo que son los autómatas y la gramática pero solamente como una breve introducción, en este caso ya vamos a verlos pero con más profundidad.

bueno primeramente, tenemos que la concatenación es asociativa, aquí nos dice que si tenemos por ejemplo 3 lenguajes, L1,L2 y L3, entonces la concatenación de L1 es equivalente a L2 y L2 es equivalente a la concatenación de L3, aunque igualmente hay caso en los que L1 puede no ser igual a L2, teniendo en consecuencia una no asociación de lenguajes debido a que ambas serian diferentes cadenas.

L1L2 diferente de L2L1
como tal un autómata nos lo tienen definido como un conjunto de estados los cuales están definidos por un número finito de símbolos de entrada o mejor conocido como alfabeto, donde tenemos variables que representan cada una el significado de esto.



en la representación finita tenemos que es un número determinado o finito de pasos a seguir para resolver el problema dado, es decir, un algoritmo con las instrucciones necesarias para resolver la situación o problema dado.

En esta parte entendemos que un autómata finito o como también es conocido “máquina de estado finito” se refiere a un modelo computacional que realiza operaciones en forma automática sobre una entrada para producir una salida, es decir, obtención de un resultado a partir de los datos ingresados o pedidos.

Este modelo está conformado como bien nos lo mencionan en los videos pasados y recientes, por un alfabeto, un conjunto de estados finitos, una función de transición, un estado inicial y un conjunto de estados finales. Su funcionamiento se basa principalmente en una función de transición, que recibe a partir de un *estado inicial* una cadena de caracteres pertenecientes al alfabeto (lo que es la entrada), y que va leyendo dicha cadena a medida que el autómata se desplaza de un estado a otro, para finalmente detenerse en un *estado final* o *de aceptación*, que representa la salida.

La finalidad de los autómatas finitos es reconocer lenguajes regulares, que corresponden a los lenguajes formales más simples según la Jerarquía de Chomsky.

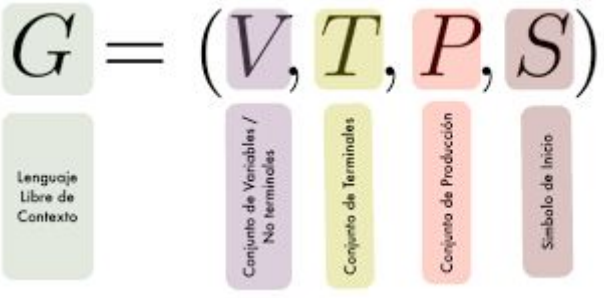
Gramática

La gramática en los autómatas se refiere básicamente a las normas que describen la secuencia de símbolos de un conjunto. Esta tiene cuatro elementos fundamentales:

G = { *NT*, *T*, *S*, *P* }

En donde:

- NT es el conjunto de elementos **No Terminales**,
- T es el conjunto de elementos **Terminales,**
- S es el **Símbolo inicial** de la gramática y finalmente,
- P es el conjunto de **Reglas de Producción**



en esta imagen podemos ver los elementos anteriormente definidos.

La gramática se puede clasificar según Padilla en 4 tipos:

1.- No restringida o recursivamente enumerables

2.- Sensible al contexto"

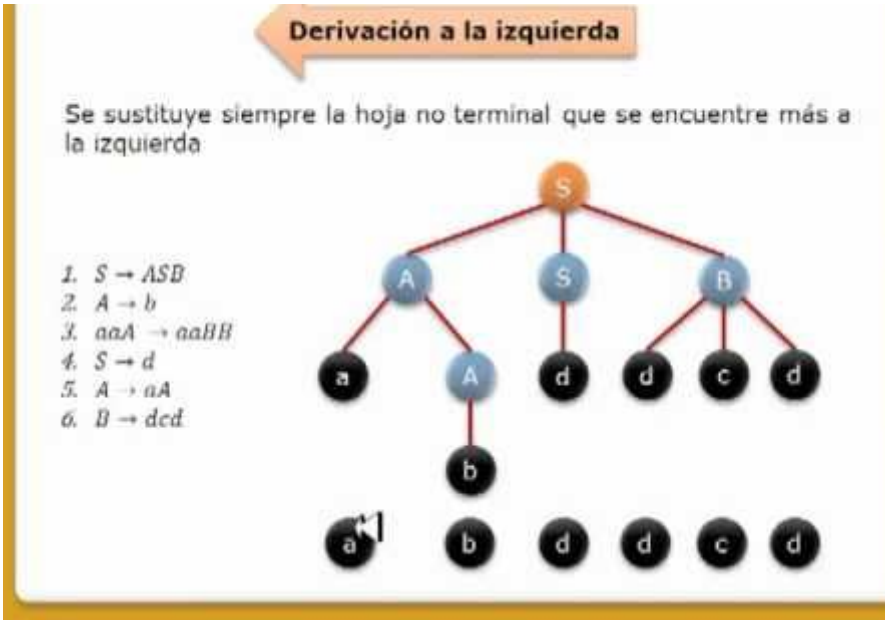
3.- libre de contexto"

4.- De contexto regular"

ÁRBOLES DE DERIVACIÓN

Finalmente tenemos el último tema que es el de los árboles de derivación. Estos nos dice que son aquellos que nos permiten mostrar gráficamente cómo podemos derivar cualquier cadena de un lenguaje a partir del símbolo distinguido de una gramática que genera ese lenguaje.

En su construcción tenemos que cada nodo del árbol va a contener un símbolo. En el nodo raíz se pone el símbolo inicial S y se efectúa una ramificación del árbol por cada producción que se aplique.



Gutierrez Arellano Rafael

¿QUÉ ES UN AUTÓMATA?

Conjunto de estados + Control → Cambio de estados en

respuesta a una entrada.

Definición formal de un autómata.
Un AF se representa como una 5-tupla: A=(Q,Σ,δ,q0,F).
DONDE:

Q = Un conjunto finitos de estados.

Σ = Un conjunto finito de símbolos de entrada (alfabeto).

q0 = El estado inicial/de comienzo.

F = Cero o más estados finales de aceptación.

δ = Función de transición. Esta solución:

- ➢ Toma un estado y un símbolo de entrada como argumentos.
- ➢ Regresa un estado
- ➢ Una "regla" de δ se escribe como δ(q,a) = p, donde **q** y **p** son estados y **a** es un estado de entrada.

REPRESENTACIÓN FINITA.

Dado un problema de programación se puede encontrar una representación finita del conjunto factible, los métodos para programas con un número finito de restricciones se pueden aplicar para resolver el problema.
Todo conjunto finito en matemáticas cuenta con un conjunto numerable los conjuntos finitos son particularmente importante en la combinación.

AUTÓMATAS FINITOS (AF).

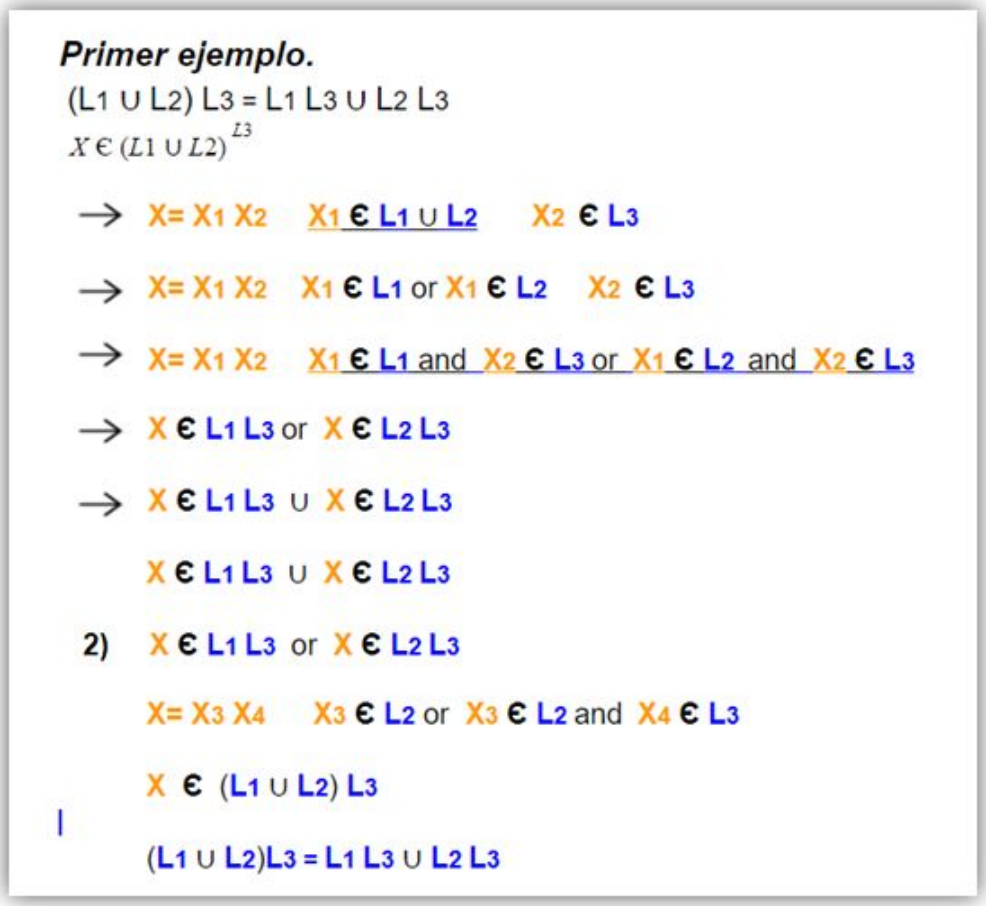
Si un **AF** está en un estado **q**, y recibe una entrada **a**, entonces el **AF** va al estado **p** (no puede ser el mismo estado **q = p**).

- ***Cada transición produce un estado.***
  Deterministas. Todos los estados son accesibles (CONEXO).
  Algoritmo para minimizar (MÍNIMO).
- ***Cada transición produce varios estados y se acepta λ.***
  No deterministas.
- ***Representación.***
  Tabla transiciones
  Diagrama
- ***Reconocer un lenguaje.***
  Aceptación.

ALGUNAS PROPIEDADES DE LENGUAJES.

1. Concatenación de lenguajes

2. La concatenación de cadenas no es conmutativa, tendríamos L1 L2 ≠ L2 L1
3. Tendríamos que L(Є) = (Є)L = **L.**
4. Propiedad distributiva.
   L1(L2 ∪ L3) = L1 L2 ∪ L1 L3
   (L1 ∪ L2) L3 = L1 L3 ∪ L2 L3



5. IF L1 ⊆ L2 and L3 ⊆ L4, then L1 L3 ⊆ L2 L4.
6. Ø* = {Є}.
7. {Є}* = {Є}.
8. IF Є Є L, then L* = L+.
9. L*L = L*L=L+.

Gramáticas

¿Qué es gramática?

Gramática en un idioma se refiere a un conjunto de reglas que se utilizan para construir oraciones en un idioma.
Existen características generales de las gramáticas de los lenguajes naturales para formalizar la nación en el contexto actual.

La forma en la cual se crea una oración de forma correcta, cuál es su contexto y cómo actúa de forma correcta.

La gramática es un ente formal para especificar, de manera finita, el conjunto de cadenas de símbolos que construye un lenguaje.

Esto es importante ya que un autómata es una construcción lógica que recibe una entrada y produce una salida en función de todo lo recibido hasta el instante, recibe una cadena de símbolos y produce una salida indicando si dicha cadena pertenece a un determinado lenguaje.

Frase/oración.
sujeto verbo objeto.
Objeto de verbo de frase sustantiva.
Sustantivo del artículo objeto del verbo.
El objeto del verbo o sustantivo.

Una gramática es una cuádrupla.

G=(VT,VN,S,P)

Donde:

VT= {conjunto finito de símbolos terminales}
VN= {conjunto finito de símbolos no terminales}
S es el símbolo inicial y pertenece a VN
P = {conjunto de producciones o reglas de derivación}

La longitud de la cadena puede ser infinita y no podemos construir infinitos estados. El lema de Pumping nos dice que no.

Gramáticas regulares -- lenguajes regulares -- autómata finito
Gramáticas libres de contexto -- lenguajes libres de contexto -- push-down automata
Gramáticas dependientes de contexto -- lenguajes dependientes de contexto -- Linear-Bounded autómata
Gramáticas sin restricciones -- lenguajes sin restricciones -- Máquina de turing

Un ejemplo para ciertas expresiones aritméticas

***G* = (*VN*, *VT*, *S*, *P*)**

- **VT={ +, -, \, *, (,), id }**
- **VN= { E}**
- **S=E**
- **P:**

1. E→E+E
2. E→E- E
3. E→(E)
4. E→ E * E
5. E→ E / E
6. E → id

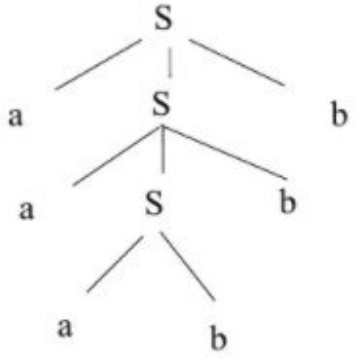
Nota: También los podemos ponerlo como una sola producción con alternativas

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

Árboles de derivación

1. Todo vértice tiene una etiqueta tomada de VT ∪ VN ∪ {***λ***}
2. La etiqueta de la raíz es el símbolo inicial *S*
3. Los vértices interiores tienen etiquetas de VN
4. Si un nodo tiene etiqueta A y $n_1 n_2 \ldots\ldots n_k$ respectivamente son hijos del vértice n, ordenados de izquierda derecha, con etiquetas $x_1, x_2 .. x_k$ respectivamente entonces:
   A ---> $x_1 x_2 .. x_k$
   debe ser una producción en P
5. Si el vértice n tiene etiqueta ***λ,*** entonces n es una hoja y es el único hijo de su padre.

- **Árbol de derivación. Ejemplo**
  - Sea *G=(VN,VT,S,P)* una GLC con P: *S→ ab|aSb*
  - La derivación de la cadena *aaabbb* será: *S ⇒ aSb ⇒ aaSbb ⇒ aaabbb* y el árbol de derivación:



Perez Armas Fausto Isaac

REPRESENTACIÓN FINITA.

Un AFD tiene un conjunto finito de estados y un conjunto finito de símbolos de entrada. El término “determinista” hace referencia al hecho de que para cada entrada sólo existe uno y sólo un estado al que el autómata puede hacer la transición a partir de su estado actual. Un estado se diseña para que sea el estado inicial, y cero o más estados para que sean estados de aceptación. Una función de transición determina cómo cambia el estado cada vez que se procesa un símbolo de entrada.

Un autómata finito determinista consta de:

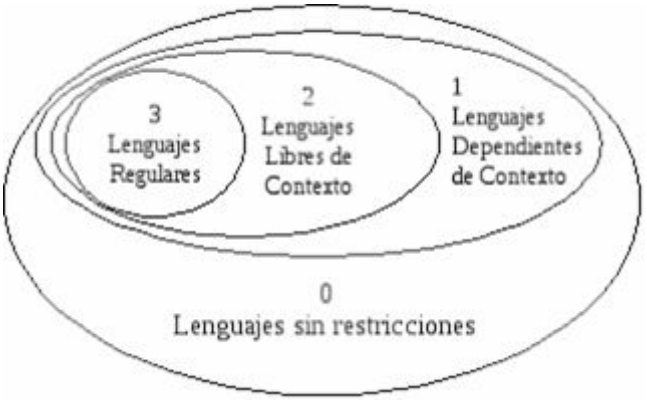
1. Un conjunto finito de estados, a menudo designado como Q.
2. Un conjunto finito de simbolos de entrada, a menudo designado como Σ
3. Una función de transición que toma como argumentos un estado y un símbolo de entrada y devuelve un estado. la funcion de transicion se designa habitualmente como δ. En nuestra representación gráfica informal del autómata, δ se ha representa mediante arcos entre los estados y las etiquetas sobre los arcos. Si q es un estado y a es un símbolo de entrada, entonces δ(q,a) es el estado p tal que existe un arco etiquetado a que va desde q hasta p.
4. Un estado inicial, uno de los estados de Q.
5. Un conjunto de estados finales o de aceptación F. El conjunto F es un subconjunto de Q.

La representación más sucinta de un AFD consiste en un listado de los cinco componentes anteriores. Normalmente, en las demostraciones, definiremos un AFD utilizando la notación de "quíntupla" siguiente: $A = (Q, \Sigma, \delta, q_0, F)$

Donde:

A Es el nombre del AFD
Q es un conjunto finito de estados
Σ son lo símbolos de entrada
q_0 es el estado inicial
$F \subseteq Q$ es el conjunto de estados finales
$\delta : Q \times V \rightarrow Q$ es la funcion de transición

GRAMÁTICA



TIPO "0" O "No restringida o recursivamente enumerables":

Incluye a todas las gramáticas formales. Estas gramáticas generan todos los lenguajes de ser reconocidos por una maquina de Turing. Y con este lenguaje se hacen los parser de un compilador

CARACTERÍSTICAS: De este tipo es que del lado derecho de cada producción puede empezar con un símbolo terminal o con un no terminal y del lado izquierdo puedes empezar con más de un símbolo no terminal.

RESTRICCIONES: Es que no tiene solamente que el del lado izquierdo debe haber por lo menos un símbolo no terminal

$$x \rightarrow y$$
$$x \in (NT/T)^+$$
$$y \in (NT/T)^*$$

NOTA: "+" significa "*sin incluir la cadena vacía"* y significa "*incluyendo la cadena vacía". "I"* significa *"o".* Estos lenguajes también son denominados "recursivamente enumerables"

TIPO "1" O "Sensible al contexto":

Estos tipos de lenguajes se resuelven mediante autómatas lineales limitados. Con este tipo se hacen los parser (analizador sintáctico) de un compilador que transforma su entrada en un árbol de derivación. El analizador sintáctico convierte el texto de entrada en otras estructuras (comúnmente árboles), que son más útiles para el posterior análisis y capturan la jerarquía implícita de la entrada

$$\alpha \rightarrow \beta \,;\, |\alpha| \leq |\beta|$$
$$\alpha = z_1 x\, z_2$$
$$\beta = z_1 y\, z_2$$
$$z_1, z_2 \in T^*$$
$$x \in NT$$
$$y \in (NT/T)^+$$

CARACTERÍSTICAS: Del lado derecho de cada producción puede empezar con un símbolo terminal o con un no terminal y del lado izquierdo puede empezar con más de un símbolo no terminal.

RESTRICCIONES: el número de no terminales del lado izquierdo de la producción debe ser menor o igual al número de símbolos del lado derecho

NOTA: Los lenguajes regulares y los libres de contexto también se puede resolver mediante autómatas lineales limitados

TIPO "2" O "Libres o Independientes de contexto":

Estos tipos de lenguajes se resuelven mediante autómatas descendentes y con este tipo de lenguaje se programa los parser en un compilador, permiten describir la mayoría de los lenguajes de programación, de hecho, la sintaxis de la mayoría de los lenguajes de programación está definida mediante gramáticas libres de contexto.

$$x \rightarrow y$$
$$x \in NT$$
$$y \in (NT/T)^*$$

CARACTERISTICAS: Del lado derecho de cada producción puede empezar con símbolo terminal o con un no terminal. Los lenguajes regulares también se pueden resolver mediante autómatas descendentes

IPO "3" O "Lenguajes regulares":

Estos tipos de lenguajes se resuelven mediante autómatas finitos y con este tipo de lenguaje se hacen los scanners. Estas gramaticas se restrigen 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 y también esta familia de lenguajes pueden ser obtenidas por medio de expresiones regulares

$$\alpha \rightarrow \beta$$
$$\alpha \in NT$$
$$\beta \in \begin{cases} aB \\ Ba \\ b \end{cases}$$
$$B \in NT$$
$$a \in T^+$$
$$b \in T^*$$

CARACTERÍSTICAS: Del lado derecho de cada producción debe empezar con un símbolo terminal. Aquí le dejo una tabla donde se representan los tipos de lenguaje según sus características.

Arboles de Derivación

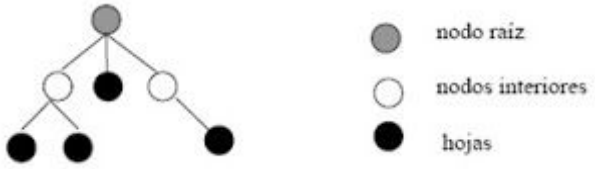
Un árbol de derivación permite mostrar gráficamente cómo se puede derivar cualquier cadena de un lenguaje a partir del símbolo distinguido de una gramática que genera ese lenguaje.

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:

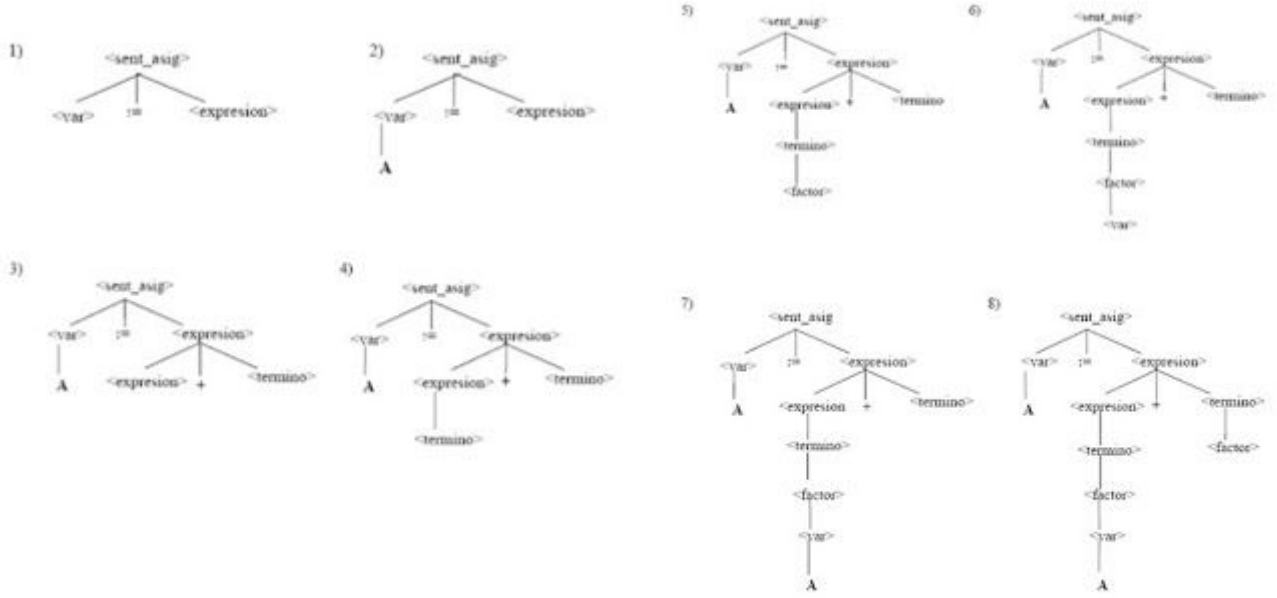
- El nodo raíz está rotulado con el símbolo distinguido de la gramática
- Cada hoja corresponde a un símbolo terminal o un símbolo no terminal;
- Cada nodo interior corresponde a un símbolo no terminal.



Para cada cadena del lenguaje generado por una gramática es posible construir (al menos) un árbol de derivación, en el cual cada hoja tiene como rótulo uno de los símbolos de la cadena.

La siguiente definición BNF describe la sintaxis (simplificada) de una sentencia de asignación de un lenguaje tipo Pascal:

Por ejemplo, la sentencia A:= A + B es una sentencia de asignación que pertenece al lenguaje definido por la definición BNF dada, y cuyo árbol de derivación se construye como se muestra a continuación:



El árbol de derivación correspondiente a la sentencia C:= D – E * F es el siguiente: