INSTITUTO TECNOLÓGICO DE IZTAPALAPA

INGENIERÍA EN SISTEMAS COMPUTACIONALES

ACTIVIDADES SEMANAES

LENGUAJES Y AUTÓMATAS I

PROFESOR:

M.C. Abiel Tomás Parra Hernández

ALUMNO:

OSORIO GARCÍA KEVIN EMMANUEL 171080231

ISC-6AM

Actividad semana 4

Introducción" y "Alfabeto, Cadenas, Lenguaje.

Cadena

Una cadena de caracteres (que también se denomina en ocasiones palabra) es una secuencia finita de símbolos seleccionados de algún alfabeto.

Una cadena o palabra es una secuencia finita de símbolos que pertenecen a un alfabeto y comúnmente se denota con la letra.

EJEMPLO: si ∑= {0,1}, entonces ∑1= {0,1}, ∑2= {00, 01, 10, 11}, ∑3= {000, 001, 010, 011, 100, 101, 110, 111}, etc.

CADENA VACÍA

La cadena vacía es aquella cadena que presenta cero apariciones de símbolos. Esta cadena, designada por £, es una cadena que puede construirse en cualquier alfabeto

EJEMPLO: observe que ∑0= {£}, independientemente de cuál sea el alfabeto ∑. Es decir, £ es la única cadena cuya longitud es 0.

Lenguajes

Un conjunto de cadenas, todas ellas seleccionadas de un ∑*, donde ∑ es un determinado alfabeto se denomina lenguaje. Ya que estas pueden ser cualquier cadena que cumpla con lo siguiente, está formada por los símbolos. Los lenguajes habituales pueden interpretarse como conjuntos de cadenas.

EJEMPLO: Seria el inglés, donde la colección de las palabras correctas inglesas es un conjunto de cadenas del alfabeto que consta de todas las letras.

EJEMPLO: Es el lenguaje C, o cualquier otro lenguaje de programación, donde los programas correctos son un subconjunto de las posibles cadenas que pueden formarse a partir del alfabeto del lenguaje.

Tipos de Lenguajes

LENGUAJES DECLARATIVOS: Es fundamentalmente lenguajes de órdenes, dominados por Sentencias que expresan “lo que hay que hacer” en vez de “cómo hacerlo”.

LENGUAJES DE ALTO NIVEL: Son los más utilizados como lenguajes de programación permiten que los algoritmos se expresen en un nivel y estilo de escritura fácilmente legible y comprensible por otros programadores.

LENGUAJE ENSAMBLADOR: Es el programa en que se realiza la tracción de un programa escrito en un programa escrito en ensamblador y lo pasa a lenguaje máquina. Directa o no directa de la traducción en que las instrucciones no son más que instrucciones que ejecuta la computadora.

LENGUAJE MAQUINA: Es como la maquina interpreta lo que nosotros queremos hacer es una lectura de 0 y 1 es decir binario.

Semana 5

REPRESENTACION FINITA, GRAMATICA Y ARBOLES DE DERIVACION.

Representación Finita.
Este modelo está conformado por un alfabeto, un conjunto de estados finito, una función de transición, un estado inicial y un conjunto de estados finales. Su funcionamiento se basa en una función de transición, que recibe a partir de un estado inicial una cadena de caracteres pertenecientes al alfabeto (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 la de reconocer lenguajes regulares, que corresponden a los lenguajes formales más simples según la Jerarquía de Chomsky.

Gramática.

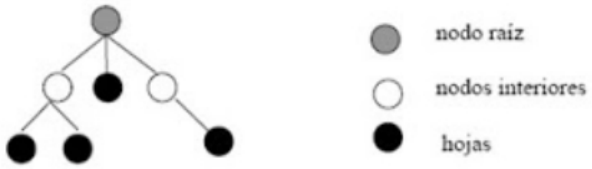
La gramática es un ente formal para especificar, de una manera finita, el conjunto de cadenas de símbolos que constituyen un lenguaje.
Es un conjunto finito de reglas que describen toda la secuencia de símbolos pertenecidas a un lenguaje especifico y dos gramáticas que describen el mismo lenguaje que llaman gramáticas equivalentes.

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.

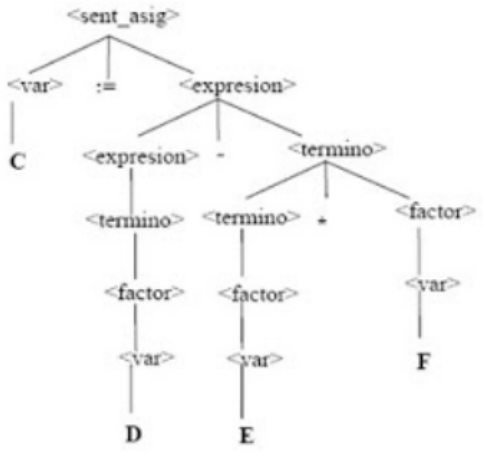
Un árbol es un conjunto de puntos, llamados nodos, unidos por líneas, llamadas arcos. Un arco conecta dos nodos distintos. Para ser un árbol un conjunto de nodos y arcos debe satisfacer ciertas propiedades:
- hay un único nodo distinguido, llamado raíz (se dibuja en la parte superior) que no tiene arcos 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.

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:



Semana 6

Gramática regular,Autómata Finito y Autómata no determinista.

Gramáticas regular

Generan los lenguajes regulares (aquellos reconocidos por un autómata finito). Son las gramáticas
más restrictivas. El lado derecho de una producción debe contener un símbolo terminal y, como
máximo, un símbolo no terminal. Estas gramáticas pueden ser:
- Lineales a derecha, si todas las producciones son de la forma

$$A \to aB \quad \text{ó} \quad A \to a \qquad \begin{cases} A \in N \cup \{S\} \\ B \in N \\ a \in T \end{cases}$$

- Lineales a izquierda, si todas las producciones son de la forma

$$A \to Ba \quad \text{ó} \quad A \to a \qquad \begin{cases} A \in N \cup \{S\} \\ B \in N \\ a \in T \end{cases}$$

En ambos casos, se puede incluir la producción S → ε, si el lenguaje que se quiere generar contiene
la cadena vacía.
Por ejemplo las siguientes gramáticas G1 y G2, son gramáticas regulares lineales a derecha y lineales
a izquierda respectivamente, que generan el lenguaje.

$$L = \{a^{2n} / n \geq 0\}$$

$G_1 = (\{A, B\}, \{a\}, P_1, S_1)$
donde P_1 es el cjto.
$S_1 \to \varepsilon$
$S_1 \to aA$
$A \to aB$
$A \to a$
$B \to aA$

$G_2 = (\{C, D\}, \{a\}, P_2, S_2)$
donde P_2 es el cjto.
$S_2 \to \varepsilon$
$S_2 \to Ca$
$C \to Da$
$C \to a$
$D \to Ca$

Autómata Finito.

Un autómata finito es un modelo matemático de una máquina que acepta cadenas de un lenguaje definido sobre un alfabeto A. Consiste en un conjunto finito de estados y un conjunto de transiciones entre esos estados, que dependen de los símbolos de la cadena de entrada. El autómata finito acepta una cadena x si la secuencia de transiciones correspondientes a los símbolos de x conduce desde el estado inicial a un estado final.

Si para todo estado del autómata existe como máximo una transición definida para cada símbolo del alfabeto, se dice que el autómata es determinístico (AFD). Si a partir de algún estado y para el mismo símbolo de entrada, se definen dos o más transiciones se dice que el autómata es no determinístico (AFND).

Formalmente un autómata finito se define como una 5-upla.

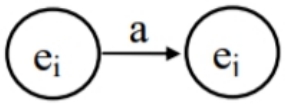
$M = <E, A, \delta, e_0, F>$ donde

E: conjunto finito de estados
A: alfabeto o conjunto finito de símbolos de entrada
δ: función de transición de estados, que se define como
- δ: E x A → E si el autómata es determinístico
- δ: E x A → P(E) si el autómata es no determinístico (P(E) es el conjunto potencia de E, es decir el conjunto de todos los subconjuntos de E)

e0: estado inicial; e0 ∈ E

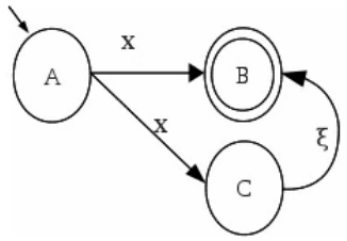
F: conjunto de estados finales o estados de aceptación; F ⊆ E.

Generalmente se asocia con cada autómata un grafo dirigido, llamado diagrama de transición de estados. Cada nodo del grafo corresponde a un estado. El estado inicial se indica mediante una flecha que no tiene nodo origen. Los estados finales se representan con un círculo doble.
Si existe una transición del estado ei al estado ej. para un símbolo de entrada a, existe entonces un arco rotulado a desde el nodo ei al nodo ej; es decir que δ(ei, a) = ej, se representa en el diagrama.



AUTÓMATA FINITO NO DETERMINISTA

Es el autómata finito que tiene transiciones vacías o que por cada símbolo desde un estado de origen se llega a más de un estado destino, es decir, es aquel que, a diferencia de los autómatas finitos deterministas, posee al menos un estado , tal que para un símbolo del alfabeto, existe más de una transición posible.



La definición formal de AFND se basa en la consideración de que a menudo según los algoritmos de transformación de expresiones y gramáticas regulares a AF terminan obteniéndose autómatas con transiciones múltiples para un mismo símbolo o transiciones vacías. Independientemente que sean indeseables, sobre todo para la implementación material, fundamentalmente mecánica, de los autómatas finitos, son imprescindibles durante la modelación de analizadores lexicográficos de los elementos gramaticales de los lenguajes de programación, llamados tókens, como literales numéricos, identificadores, cadenas de texto, operadores, etc.

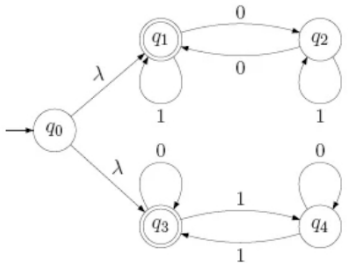
Los AFND sondefiniciones no tan deseables dentro de los lenguajes regulares porquedificultan su implementación tanto mecánica como informática; aunque en la mayoría de las transformaciones a lo interno de los LR (expresiones regulares a AF, gramáticas regulares a AF) conducen a AFND. Los AFND, por tanto, son imprescindibles en el análisis lexicográfico y el diseño de los lenguajes de programación.

Haciendo la analogía con los AFDs, en un AFND puede darse cualquiera de estos dos casos:

- Que existan transiciones del tipo δ(q,a)=q1 y δ(q,a)=q2 siendo q1≠q2 ;

-Que existan transiciones del tipo ,δ(q,ε) siendo un estado no-final, o bien un estado final pero con transiciones hacia otros estados.

Cuando se cumple el segundo caso, se dice que el autómata es un autómata finito no determinista con transiciones vacías o transiciones ε (abreviado AFNDε). Estas transiciones permiten al autómata cambiar de estado sin procesar ningún símbolo de entrada.



Semana 7

Autómatas Finitos Deterministas (AFD)

Este tipo de autómatas admite su definición de dos maneras bien diferentes:: Como autómatas traductores o reconocedores. La definición como autómatas traductores continua a la definición de las máquinas secuenciales, y se los podría definir como una subclase de estas, ya que los autómatas finitos tendrían como limitante no poder iniciar desde cualquier estado como lo hacen en las máquinas secuenciales.

La forma que adoptaremos para la definición de los autómatas finitos deterministas es como autómatas reconocedores, ya que se ajusta con los contenidos de la informática teórica y utilización que se les da dentro del diseño de los analizadores léxicos.

Estos autómatas solo se limitarán a aceptar o no una determinada cadena recibida en la entrada, por lo tanto podemos decir que la salida de los mismos solo tendrá dos valores posibles aceptar o no aceptar a la palabra de entrada.

Al igual que en las máquinas secuenciales, estos autómatas transitarán entre un conjunto finito de estados posibles, a medida que reciban sucesivamente los caracteres de entrada, en un instante determinado de tiempo el autómata solo podrá estar en uno y solo uno de los estados posibles.

Una característica importante de este tipo de autómatas es el determinismo, lo cuál significa que estando en un estado y recibiendo una entrada del exterior el autómata tendrá la posibilidad de transitar a uno y solo un estado del conjunto de estados posibles.

Con respecto al conjunto de estados ahora se pueden clasificar en tres tipos: Estado Inicial, que es pon donde comenzará la ejecución de la máquina; Estados finales o de aceptación que será un subconjunto del conjunto de estados por los que transitará la máquina, y si cuando se hayan terminado de procesar todos los símbolos de entrada y no reste ningún símbolo por leer, la máquina quede posicionada en uno de estos estados de aceptación, se concluirá que la cadena procesada será aceptada por el autómata. y Estados Intermedios, que tienen comportamiento idéntico a los definidos en las máquinas secuenciales.

Los autómatas finitos deterministas quedarán formalmente definida mediante una quíntupla como sigue:

$$AFD = (\sum , Q, q_0, F, f) \text{ donde:}$$

$\sum$	Alfabeto de símbolos de entrada.
Q	Conjunto finito de estados
q_0	$q_0 \in Q$ – estado inicial previsto
F	$F \subseteq Q$ - es el conjunto de estado finales de aceptación.
f	Función de transición de estados definida como **f: Q x** $\sum$ ——— **Q**

Teorema de Myhill-Nerode

Si L es un lenguaje regular, entonces, por definición, hay un DFA A que lo reconoce, con solo un número finito de estados. Si hay n estados, entonces particione el conjunto de todas las cadenas finitas en n subconjuntos, donde el subconjunto S i es el conjunto de cadenas que, cuando se dan como entrada al autómata A , hacen que termine en el estado i .

Por cada dos cadenas X y Y que pertenecen al mismo subconjunto, y para cada opción de una tercera cadena z , autómata A alcanza el mismo estado en la entrada XZ , ya que llega en la entrada yz , y por lo tanto debe aceptar ya sea tanto de las entradas xz e yz o rechazar ambos. Por lo tanto, ninguna cadena z puede ser una extensión distintiva de x y y , así que deben estar relacionados por R L.

Por lo tanto, s i es un subconjunto de una clase de equivalencia de R L. Combinando este hecho con el hecho de que cada miembro de una de estas clases de equivalencia pertenece a uno de los conjuntos S i , esto da una relación de muchos a uno de los estados de A a las clases de equivalencia, lo que implica que el número de clases de equivalencia es finito y como máximo n.

En la otra dirección, suponga que R L tiene un número finito de clases de equivalencia. En este caso, es posible diseñar un autómata finito determinista que tenga un estado para cada clase de equivalencia.

El estado de inicio del autómata corresponde a la clase de equivalencia que contiene la cadena vacía, y la función de transición de un estado X en el símbolo de entrada y lleva al autómata a un nuevo estado, el estado correspondiente a la clase de equivalencia que contiene la cadena xy , donde x es una cadena arbitrariamente elegido en la clase de equivalencia de X . La definición de la relación Myhill-Nerode implica que la función de transición está bien definida: no importa qué cadena representativa x se elija para el estado X , se obtendrá el mismo valor de función de transición.

Un estado de este autómata es aceptable si la clase de equivalencia correspondiente contiene una cadena en L ; en este caso, nuevamente, la definición de la relación implica que todas las cadenas de la misma clase de equivalencia también deben pertenecer a L , porque de lo contrario la cadena vacía sería una cadena distintiva para algunos pares de cadenas de la clase. Así, la existencia de un autómata finito que reconoce L implica que la relación Myhill-Nerode tiene un número finito de clases de equivalencia, a lo sumo igual al número de estados del autómata, y la existencia de un número finito de clases de equivalencia implica la existencia de un autómata con tantos estados.